LogiCORE IP Image Statistics v3.0

Product Guide

PG008 October 19, 2011

XILINX®

Table of Contents

LogiCORE IP Image Statistics v3.0

Introduction

The Xilinx LogiCORE™ IP Image Statistics core implements the computationally intensive metering functionality common in digital cameras, camcorders and imaging devices. This core generates a set of statistics for color histograms, mean and variance values, edge and frequency content for 16 user-defined zones on a per frame basis. The statistical information collected may be used in the control algorithms for Auto-Focus, Auto-White Balance, and Auto-Exposure for image processing applications.

Features

- High-definition (1080p60) resolutions
- Up to 4096 total pixels and 4096 total rows
- Selectable processor interface: EDK pCore or General Purpose Processor
- 16 programmable zones
- 8, 10, or 12-bit input precision
- Outputs for all zones and color channels:
  - Minimum and maximum color values
  - Sum and sum of squares for each color value
  - Low and high frequency content
  - Horizontal, vertical and diagonal edge content
- Outputs for pre-selected zone(s):
  - Y channel histogram
  - R,G,B channel histograms
  - Two-dimensional Cr-Cb histogram

LogiCORE IP Facts Table					
Core Specifics					
Supported Device Family(1)	Virtex®-7, Kintex™-7, Virtex-6, Spartan®-6				
Supported User Interfaces	General Purpose Processor Interface, AXI4-Lite				
	Resources(2)				**Frequency**
Configuration	**LUTs**	**FFs**	**DSP Slices**	**Block RAMs(3)**	**Max. Freq.**
Data Width=8	2821	3777	16	7(18)+1(36)	202
Data Width=10	3077	4129	16	1(18)+7(36)	202
Data Width=12	3363	4481	16	2(18)+17(36)	202
Provided with Core					
Design Files	Netlists, EDK pCore files, C drivers				
Example Design	Not Provided				
Test Bench	VHDL(4)				
Constraints File	Not Provided				
Simulation Model	VHDL or Verilog Structural Model C Model(4)				
Tested Design Tools					
Design Entry Tools	CORE Generator™, ISE® 13.3, Platform Studio (XPS)				
Simulation(5)	Mentor Graphics ModelSim, QuestaSim, ISim				
Synthesis Tools	XST 13.3				
Support					
Provided by Xilinx, Inc.					

1. For a complete listing of supported devices, see the release notes for this core.
2. Resources listed are for Virtex-6 devices, with an EDK pCore interface, and Maximum Number of Columns and Rows of 2200. For more complete device performance numbers, see Resource Utilization, page 9
3. Indicating the number of RAMB18E1 and RAMB36E1 primitives used.
4. HDL test bench and C Model available on the Image Statistics product page.
5. For the supported versions of the tools, see the ISE Design Suite 13: Release Notes Guide.

Chapter 1

Overview

Most digital cameras implement a form of automatic white balance (AWB) and automatic exposure (AE) control. Cameras with an adjustable focus (AF) lens also implement automatic focus. All of these algorithms have the need for sophisticated image statistics gathered from the image or video to process and implement controls and algorithms for these functions. Generating image statistics is a data intensive process, and the Image Statistics core provides an efficient, programmable solution. The resulting image statistics provide the basic data for software based AE, AWB and AF algorithms. The Image Statistics core supports multi-zone metering on rectangular regions, and 16 software defined zones, as shown in Figure 1-1.

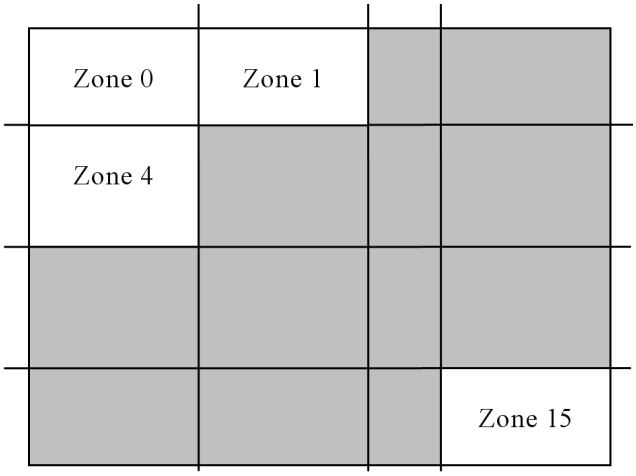


Figure 1-1: **16 Zone Metering**

Minimum, maximum, sum, and sum of squares values are calculated for all color channels for all 16 zones. Frequency and edge content is also calculated for all zones in luminance values provided by an RGB-to-YCrCb converter internal to the Image Statistics core. The RGB, Y, and two-dimensional Cr-Cb histograms are calculated over predefined sets of zones.

Standards Compliance

The Image Statistics core is compliant with the AXI4-Lite interconnect standard as defined in UG761, *AXI Reference Guide*.

Feature Summary

The Image Statistics core provides data from video streams of a maximum resolution of 4096 columns by 4096 rows and supports the bandwidth necessary for high-definition (1080p60) resolutions. The core generates data from the video stream in the form of minimum and maximum color values, sum and sum of squares for each color value, low and high frequency content and horizontal, vertical and edge content from 16 programmable zones. The core can also generate histograms for RGB channels, Y channel and two dimensional Cr-Cb channels.

The core can be generated as an EDK pCore (AXI4-Lite interconnect) or with a generic General Purpose Processor interface where all the user register connections are exposed as ports to the core.

Applications

- Automatic Exposure (AE) control
- Automatic Sensor Gain (AG) control
- Auto Focus (AF) control of the lens assembly
- Digital contrast/brightness adjustment
- Global histogram equalization
- White Balance correction

Licensing

The Image Statistics core provides the following three licensing options:

- Simulation Only
- Full System Hardware Evaluation
- Full

After installing the required Xilinx ISE software and IP Service Packs, choose a license option.

Simulation Only

The Simulation Only Evaluation license key is provided with the Xilinx CORE Generator tool. This key lets you assess core functionality with either the example design provided with the Image Statistics core, or alongside your own design and demonstrates the various interfaces to the core in simulation. (Functional simulation is supported by a dynamically generated HDL structural model.)

No action is required to obtain the Simulation Only Evaluation license key; it is provided by default with the Xilinx CORE Generator software.

Full System Hardware Evaluation

The Full System Hardware Evaluation license is available at no cost and lets you fully integrate the core into an FPGA design, place-and-route the design, evaluate timing, and perform functional simulation of the Image Statistics core using the example design and demonstration test bench provided with the core.

In addition, the license key lets you generate a bitstream from the placed and routed design, which can then be downloaded to a supported device and tested in hardware. The core can be tested in the target device for a limited time before timing out (all measured results remain in a cleared state), at which time it can be reactivated by reconfiguring the device.

To obtain a Full System Hardware Evaluation license, do the following:

1. Navigate to the product page for this core.
2. Click Evaluate.
3. Follow the instructions to install the required Xilinx ISE software and IP Service Packs.

The timeout period for this core is set to approximately eight hours for a 74.25 MHz clock. Using a faster or slower clock will change the timeout period proportionally. For example, using a 150 MHz clock will result in a timeout period of approximately four hours.

Full

The Full license key is available when you purchase the core and provides full access to all core functionality both in simulation and in hardware, including:

- Functional simulation support
- Full implementation support including place and route and bitstream generation
- Full functionality in the programmed device with no time outs

To obtain a Full license key, you must purchase a license for the core. Click on the "Order" link on the Xilinx.com IP core product page for information on purchasing a license for this core. After doing so, click the "How do I generate a license key to activate this core?" link on the Xilinx.com IP core product page for further instructions.

Installing Your License File

The Simulation Only Evaluation license key is provided with the ISE CORE Generator system and does not require installation of an additional license file. For the Full System Hardware Evaluation license and the Full license, an email will be sent to you containing instructions for installing your license file. Additional details about IP license key installation can be found in the ISE Design Suite Installation, Licensing and Release Notes document.

Performance

The following sections detail the performance characteristics of the Image Statistics core.

Maximum Frequencies

The following are typical clock frequencies for the target devices. The maximum achievable clock frequency can vary. The maximum achievable clock frequency and all resource counts can be affected by other tool options, additional logic in the FPGA device, using a different version of Xilinx tools and other factors.

- Virtex®-7 FPGA: 202 MHz
- Kintex™-7 FPGA: 202 MHz
- Virtex-6 FPGA: 202 MHz

- Spartan®-6 FPGA: 159 MHz

Throughput

The Image Statistics core produces statistics in real time for the data it consumes. If timing constraints are met, the throughput is equal to the rate at which video data is written into the core. In numeric terms, 1080P/60 RGB represents an average data rate of 124.4 Mpixels/sec or a burst data rate of 148.5 Mpixels/sec.

Resource Utilization

For an accurate measure of the usage of device resources for a particular instance, click **View Resource Utilization** in CORE Generator after generating the core.

Information presented in Table 1-1 through Table 1-4 show guidelines to the resource utilization of the Image Statistics core for Virtex®-6, Kintex-7,Virtex-6, and Spartan®-6 FPGA families. The design was tested using Xilinx ISE v13.3 tools with area constraints (see table footnotes) and default tool options.

Table 1-1 through Table 1-4 present the resource utilization and target clock frequencies of the Image Statistics core for all input width combinations with three typical values for Maximum Number of Rows and Maximum Number of Columns. These characterization tests were performed with the Maximum Number of Rows and Maximum Number of Columns parameters set to equal values, all histograms enabled and using the General Purpose Processor interface.

Table 1-1: **Resource Utilization and Target Speed for Virtex-7 Devices**

Data Width	Max Rows Max Cols	Histogram			LUTs	FFs	RAM36	RAM18	DSP48s	Clock F_{MAX} (MHz)
		Y	RGB	CC						
8	1023	Yes	Yes	Yes	2624	3514	0	6	16	202
8	2200	Yes	Yes	Yes	2828	3777	1	7	16	202
10	1023	Yes	Yes	Yes	2868	3866	6	0	16	202
10	2200	Yes	Yes	Yes	3137	4129	7	1	16	202
12	1023	Yes	Yes	Yes	3118	4218	11	5	16	202
12	2200	Yes	Yes	Yes	3326	4481	17	2	16	202

Device and speed file: XC7V330T-1 (ADVANCED 1.02 2011-09-27)

Table 1-2: **Resource Utilization and Target Speed for Kintex-7 Devices**

Data Width	Max Rows Max Cols	Histogram			LUTs	FFs	RAM36	RAM18	DSP48s	Clock F_{MAX} (MHz)
		Y	RGB	CC						
8	1023	Yes	Yes	Yes	2570	3514	0	6	16	202
8	2200	Yes	Yes	Yes	2791	3777	1	7	16	202
10	1023	Yes	Yes	Yes	2833	3866	6	0	16	202
10	2200	Yes	Yes	Yes	3052	4129	7	1	16	202
12	1023	Yes	Yes	Yes	3028	4218	11	5	16	202

Table 1-2: **Resource Utilization and Target Speed for Kintex-7 Devices *(Cont'd)***

Data Width	Max Rows Max Cols	Histogram			LUTs	FFs	RAM36	RAM18	DSP48s	Clock F_{MAX} (MHz)
		Y	RGB	CC						
12	2200	Yes	Yes	Yes	3355	4481	17	2	16	202

Device and speed file: XC7K70T-1 (ADVANCED 1.02 2011-09-27)

Table 1-3: **Resource Utilization and Target Speed for Virtex-6 Devices**

Data Width	Max Rows Max Cols	Histogram			LUTs	FFs	RAM36	RAM18	DSP48s	Clock F_{MAX} (MHz)
		Y	RGB	CC						
8	1023	Yes	Yes	Yes	2583	3514	0	6	16	202
8	2200	Yes	Yes	Yes	2821	3777	1	7	16	202
10	1023	Yes	Yes	Yes	2871	3866	6	0	16	202
10	2200	Yes	Yes	Yes	3077	4129	7	1	16	202
12	1023	Yes	Yes	Yes	3153	4218	11	5	16	202
12	2200	Yes	Yes	Yes	3363	4481	17	2	16	202

Device and speed file: XC6VLX75T,-1 (PRODUCTION 1.15 2011-09-27)

Table 1-4: **Resource Utilization and Target Speed for Spartan-6 Devices**

Data Width	Max Rows Max Cols	Histogram			LUTs	FFs	RAM36	RAM18	DSP48s	Clock F_{MAX} (MHz)
		Y	RGB	CC						
8	1023	Yes	Yes	Yes	4254	3528	1	5	16	158
8	2200	Yes	Yes	Yes	4557	3811	4	5	16	158
10	1023	Yes	Yes	Yes	4715	4002	6	6	16	158
10	2200	Yes	Yes	Yes	5086	4288	10	5	16	157
12	1023	Yes	Yes	Yes	5053	4358	26	1	16	156
12	2200	Yes	Yes	Yes	5472	4644	36	0	16	158

Device and speed file: XC6SLX25-2 (PRODUCTION 1.20 2011-09-27)

Optional Histogram Calculation

Table 1-5 through Table 1-8 present resource utilization numbers for all possible combinations of optional histogram settings, for all supported device families, using 8-bit input data and the maximum numbers of rows and columns set to 2200.

Table 1-5: **Optional Histogram Calculation: Resource Utilization Virtex-7 Devices (XC7V330T-1)**

Data Width	MAX Rows MAX Cols	Histogram			LUTs	FF	RAM36	RAM18	DSP 48s	Clock F_{MAX} (MHz)
		Y	RGB	CC						
8	2200	Yes	Yes	Yes	2828	3777	1	7	16	202
8	2200	Yes	Yes	No	2787	3681	1	6	16	202
8	2200	Yes	No	Yes	2677	3591	1	6	14	202
8	2200	Yes	No	No	2582	3474	1	5	14	202

Table 1-5: **Optional Histogram Calculation: Resource Utilization Virtex-7 Devices (XC7V330T-1) *(Cont'd)***

Data Width	MAX Rows MAX Cols	Histogram			LUTs	FF	RAM36	RAM18	DSP 48s	Clock F_{MAX} (MHz)
		Y	RGB	CC						
8	2200	No	Yes	Yes	2617	3444	1	4	16	202
8	2200	No	Yes	No	2498	3348	1	3	16	202
8	2200	No	No	Yes	2402	3258	1	3	14	202
8	2200	No	No	No	2372	3137	1	2	14	202

Table 1-6: **Optional Histogram Calculation: Resource Utilization Kintex-7 Devices (XC7K70T-1)**

Data Width	MAX Rows MAX Cols	Histogram			LUTs	FF	RAM36	RAM18	DSP 48s	Clock F_{MAX} (MHz)
		Y	RGB	CC						
8	2200	Yes	Yes	Yes	2791	3777	1	7	16	202
8	2200	Yes	Yes	No	2709	3681	1	6	16	202
8	2200	Yes	No	Yes	2562	3591	1	6	14	202
8	2200	Yes	No	No	2537	3474	1	5	14	202
8	2200	No	Yes	Yes	2594	3444	1	4	16	202
8	2200	No	Yes	No	2467	3348	1	3	16	202
8	2200	No	No	Yes	2384	3258	1	3	14	202
8	2200	No	No	No	2312	3137	1	2	14	202

Table 1-7: **Optional Histogram Calculation: Resource Utilization Virtex-6 Devices (XC6VLX75T-1)**

Data Width	MAX Rows MAX Cols	Histogram			LUTs	FF	RAM36	RAM18	DSP 48s	Clock F_{MAX} (MHz)
		Y	RGB	CC						
8	2200	Yes	Yes	Yes	2821	3777	1	7	16	202
8	2200	Yes	Yes	No	2741	3681	1	6	16	202
8	2200	Yes	No	Yes	2661	3591	1	6	14	202
8	2200	Yes	No	No	2479	3474	1	5	14	202
8	2200	No	Yes	Yes	2561	3444	1	4	16	202
8	2200	No	Yes	No	2542	3348	1	3	16	202
8	2200	No	No	Yes	2464	3258	1	3	14	202
8	2200	No	No	No	2309	3137	1	2	14	202

Table 1-8: **Optional Histogram Calculation: Resource Utilization Spartan-6 Devices (XC6SLX25-1)**

Data Width	MAX Rows MAX Cols	Histogram			LUTs	FF	RAM36	RAM18	DSP 48s	Clock F_{MAX} (MHz)
		Y	RGB	CC						
8	2200	Yes	Yes	Yes	4557	3811	4	5	16	158
8	2200	Yes	Yes	No	4433	3711	4	4	16	159
8	2200	Yes	No	Yes	4279	3621	4	4	14	158
8	2200	Yes	No	No	4096	3505	4	3	14	158
8	2200	No	Yes	Yes	4108	3472	4	2	16	130
8	2200	No	Yes	No	3882	3377	4	1	16	157
8	2200	No	No	Yes	2440	3277	4	1	14	160
8	2200	No	No	No	3542	3161	4	0	14	157

XILINX®

Chapter 2

Core Interfaces and Register Space

This chapter provides detailed descriptions for each interface. In addition, detailed information about configuration and control registers is included.

Port Descriptions

Processor Interfaces

Processor interfaces provide the system designer with the ability to dynamically control the parameters within the core. The Image Statistics core supports two processor interface options:

- EDK pCore Interface
- General Purpose Processor Interface

The Xilinx Streaming Video Interface is a set of signals common to both interface options and to all video Image Processing (iPipe) cores. It is described in Table 2-1 along with more detailed descriptions of the ports following the table.

Table 2-1: **Port Descriptions for the Xilinx Streaming Video Interface**

Port Name	Port Width	Direction	Description
video_data_in	3DATA_WIDTH	input	Data input bus
hblank_in	1	input	Horizontal blanking input
vblank_in	1	input	Vertical blanking input
active_video_in	1	input	Active video signal input

- **video_data_in:** This bus contains the video input in `DATA_WIDTH` bits wide unsigned integer representation.

Table 2-2: **Video Data In Width**

	Red	Blue	Green
Bits	3DATA_WIDTH-1: 2DATA_WIDTH	2DATA_WIDTH-1: DATA_WIDTH	DATA_WIDTH-1:0
Video Data Signals	R	B	G

- **hblank_in:** The `hblank_in` signal conveys information about the blank/non-blank regions of video scan lines.
- **vblank_in:** The `vblank_in` signal conveys information about the blank/non-blank regions of video frames, and is used by the Image Statistics core to detect the end of a

frame, when user registers can be copied to active registers to avoid visual tearing of the image.

- **active_video_in:** The `active_video_in` signal is high when valid data is presented at the input.

Xilinx Streaming Video Interface

The Xilinx Streaming Video Interface (XSVI) is a set of signals that is used to stream video data between video IP cores. XSVI is also defined as an Embedded Development Kit (EDK) bus type so that the tool can automatically create input and output connections to the core. This definition is embedded in the pCore interface provided with the IP, and it allows an easy way to cascade connections of Xilinx Video Cores. The Image Statistics IP core uses the following subset of the XSVI signals:

- `video_data`
- `vblank`
- `hblank`
- `active_video`

Other XSVI signals on the XSVI input bus, such as `video_clk`, `vsync`, `hsync`, `field_id`, and `active_chr` do not affect the function of this core.

Note: These signals are neither propagated, nor driven on the XSVI output of this core.

The following is an example EDK Microprocessor Peripheral Definition (.MPD) file definition.

Input Side:

```
BUS_INTERFACE BUS = XSVI_STATISTICS,  BUS_TYPE = TARGET, BUS_STD =
XSVI

PORT hblank_i = hblank,            DIR=I,   BUS=XSVI_STATISTICS

PORT vblank_i = vblank,            DIR=I,   BUS=XSVI_STATISTICS

PORT active_video_i = active_video,DIR=I,   BUS=XSVI_STATISTICS

PORT video_data_i=video_data,DIR=I,VEC=[C_DATA_WIDTH1:0],BUS=XSVI_
STATISTICS
```

The Image Statistics IP core is fully synchronous to the core clock, `clk`. Consequently, the input XSVI bus is expected to be synchronous to the input clock, `clk`. The `video_clk` signal of the input is not used.

EDK pCore Interface

Many imaging applications utilize an embedded processor to dynamically control parameters within IP cores. The EDK pCore Interface generates AXI4-Lite Bus interface ports in addition to the Xilinx Streaming Video Interface, `clk`, `ce`, and `sclr` signals. Signal details of the AXI4-Lite interface are shown in Table 2-3. The AXI4-Lite bus signals are automatically connected when the generated pCore is inserted into an EDK project.

Table 2-3: **AXI4-Lite Interface Pinout**

Pin Name	Direction	Width	Description
AXI Global System Signals[(1)]			
S_AXI_ACLK	I	1	AXI Clock

Table 2-3: **AXI4-Lite Interface Pinout *(Cont'd)***

Pin Name	Direction	Width	Description
S_AXI_ARESETN	I	1	AXI Reset, active Low
IP2INTC_Irpt	O	1	Interrupt request output
AXI Write Address Channel Signals(1)			
S_AXI_AWADDR	I	[(C_S_AXI_ADDR_WIDTH-1):0]	AXI4-Lite Write Address Bus. The write address bus gives the address of the write transaction.
S_AXI_AWVALID	I	1	AXI4-Lite Write Address Channel Write Address Valid. This signal indicates that valid write address is available. • 1 = Write address is valid. • 0 = Write address is not valid.
S_AXI_AWREADY	O	1	AXI4-Lite Write Address Channel Write Address Ready. Indicates core is ready to accept the write address. • 1 = Ready to accept address. • 0 = Not ready to accept address.
AXI Write Data Channel Signals(1)			
S_AXI_WDATA	I	[(C_S_AXI_DATA_WIDTH-1):0]	AXI4-Lite Write Data Bus.
S_AXI_WSTRB	I	[C_S_AXI_DATA_WIDTH/8-1:0]	AXI4-Lite Write Strobes. This signal indicates which byte lanes to update in memory.
S_AXI_WVALID	I	1	AXI4-Lite Write Data Channel Write Data Valid. This signal indicates that valid write data and strobes are available. • 1 = Write data/strobes are valid. • 0 = Write data/strobes are not valid.
S_AXI_WREADY	O	1	AXI4-Lite Write Data Channel Write Data Ready. Indicates core is ready to accept the write data. • 1 = Ready to accept data. • 0 = Not ready to accept data.
AXI Write Response Channel Signals(1)			
S_AXI_BRESP(2)	O	[1:0]	AXI4-Lite Write Response Channel. Indicates results of the write transfer. • 00b = OKAY - Normal access has been successful. • 01b = EXOKAY - Not supported. • 10b = SLVERR - Error. • 11b = DECERR - Not supported.
S_AXI_BVALID	O	1	AXI4-Lite Write Response Channel Response Valid. Indicates response is valid. • 1 = Response is valid. • 0 = Response is not valid.

Table 2-3: **AXI4-Lite Interface Pinout *(Cont'd)***

Pin Name	Direction	Width	Description
S_AXI_BREADY	I	1	AXI4-Lite Write Response Channel Ready. Indicates Master is ready to receive response. • 1 = Ready to receive response. • 0 = Not ready to receive response.
AXI Read Address Channel Signals[1]			
S_AXI_ARADDR	I	[(C_S_AXI_ADDR_WIDTH-1):0]	AXI4-Lite Read Address Bus. The read address bus gives the address of a read transaction.
S_AXI_ARVALID	I	1	AXI4-Lite Read Address Channel Read Address Valid. • 1 = Read address is valid. • 0 = Read address is not valid.
S_AXI_ARREADY	O	1	AXI4-Lite Read Address Channel Read Address Ready. Indicates core is ready to accept the read address. • 1 = Ready to accept address. • 0 = Not ready to accept address.
AXI Read Data Channel Signals[1]			
S_AXI_RDATA	O	[(C_S_AXI_DATA_WIDTH-1):0]	AXI4-Lite Read Data Bus.
S_AXI_RRESP[2]	O	[1:0]	AXI4-Lite Read Response Channel Response. Indicates results of the read transfer. • 00b = OKAY - Normal access has been successful. • 01b = EXOKAY - Not supported. • 10b = SLVERR - Error. • 11b = DECERR - Not supported.
S_AXI_RVALID	O	1	AXI4-Lite Read Data Channel Read Data Valid. This signal indicates that the required read data is available and the read transfer can complete. • 1 = Read data is valid. • 0 = Read data is not valid.
S_AXI_RREADY	I	1	AXI4-Lite Read Data Channel Read Data Ready. Indicates master is ready to accept the read data. • 1 = Ready to accept data. • 0 = Not ready to accept data.

1. The function and timing of these signals are defined in the *AMBA AXI Protocol Version: 2.0 Specification*.
2. For signals S_AXI_RRESP[1:0] and S_AXI_BRESP[1:0], the core does not generate the Decode Error ('11') response. Other responses like '00' (OKAY) and '10' (SLVERR) are generated by the core based upon certain conditions.

The core symbol diagram for the AXI4-Lite Interface is shown in Figure 2-1.

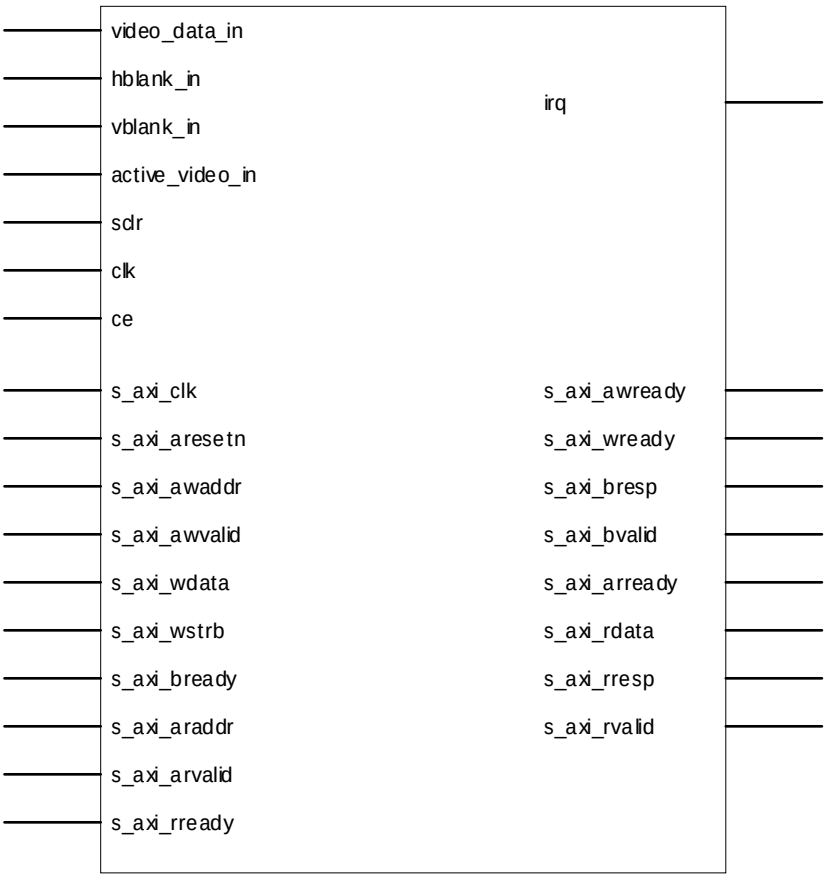


Figure 2-1: **Core Symbol**

Register Space

Generating the Image Statistics core with an EDK pCore interface provides a memory-mapped interface for the programmable registers within the core, described in Table 2-4.

All of the registers are readable, enabling verification of written values or read back of current values.

Table 2-4: **EDK pCore Interface Register Description**

Address Offset BASEADDR+	Register Name	Access Type	Default Value	Description
0x000	stats_reg_00_control	R/W	1	Control register (Table 2-5) Bit 0: SW_ENABLE Bit 1: REG_UPDATE Bit 2: READOUT Bit 3: CLR_STATUS
0x004	stats_reg_01_sw_reset	R/W	0	Bit 1: SW_RESET (1: reset, 0: not reset)

Table 2-4: **EDK pCore Interface Register Description *(Cont'd)***

Address Offset BASEADDR+	Register Name	Access Type	Default Value	Description
0x008	stats_reg_02_status	R	0	General status register (Table 2-6) Bit 0: VSYNC Bit 1: DONE Frame acquisition complete Bit 2: VBLANK_ERROR Bit 3: HBLANK_ERROR
0x00c	stats_reg_03_irq_control	R/W	258	Bit 0: VSYNC Interrupt enable Bit 1: DONE Interrupt enable Bit 2: VBLANK_ERROR Interrupt enable Bit 3: HBLANK_ERROR Interrupt enable Bit 8: General Interrupt Enable
0x010	stats_reg_04_hmax0	R/W	¼ MAX_COLS	Position of the first vertical zone delimiter
0x014	stats_reg_05_hmax1	R/W	½ MAX_COLS	Position of the second vertical zone delimiter
0x018	stats_reg_06_hmax2	R/W	¾ MAX_COLS	Position of the third vertical zone delimiter
0x01c	stats_reg_07_vmax0	R/W	¼ MAX_ROWS	Position of the first horizontal zone delimiter
0x020	stats_reg_08_vmax1	R/W	½ MAX_ROWS	Position of the second horizontal zone delimiter
0x024	stats_reg_09_vmax2	R/W	¾ MAX_ROWS	Position of the third horizontal zone delimiter
0x028	stats_reg_10_hist_zoom_factor	R/W	0	Bit 0-1 control CrCb histogram zooming: 00: No zoom, full Cb and Cr range 01: Zoom by 2 10: Zoom by 4 11: Zoom by 8
0x02c	stats_reg_11_rgb_hist_zone_en	R/W	65535	Bits 0-15 correspond to zones 0-15, enabling RGB histogramming for the selected zones
0x030	stats_reg_12_ycc_hist_zone_en	R/W	65535	Bits 0-15 correspond to zones 0-15, enabling Y and CrCb histogramming for the selected zones
0x034	stats_reg_13_zone_addr	R/W	0	Bits 0-3 select a zone for readout
0x038	stats_reg_14_color_addr	R/W	0	Bits 0-1 select a color channel for readout 00: Red 01: Green 1X: Blue
0x03c	stats_reg_15_hist_addr	R/W	0	Bits 0-[DATA_WIDTH-1] address histograms
0x040	stats_reg_16_addr_valid	R/W	0	Bit 0 qualifies zone_addr, color_addr and hist_addr
0x044	stats_reg_17_data_valid	R	0	Bit 0 qualifies valid data on core outputs corresponding to addr inputs

Table 2-4: **EDK pCore Interface Register Description *(Cont'd)***

Address Offset BASEADDR+	Register Name	Access Type	Default Value	Description
0x050	stats_reg_20_max	R	0	Maximum value measured for the currently selected zone and color channel
0x054	stats_reg_21_min	R	0	Minimum value measured for the currently selected zone and color channel
0x058	stats_reg_22_sum_lo	R	0	Higher and Lower 32 bits of the sum of values for the currently selected zone and color channel
0x005c	stats_reg_23_sum_hi	R	0	
0x060	stats_reg_24_pow_lo	R	0	Higher and Lower 32 bits of the sum of squared values for the currently selected zone and color channel
0x064	stats_reg_25_pow_hi	R	0	
0x068	stats_reg_26_hsobel_lo	R	0	Higher and lower 32 bits of the sum of absolute values of the Horizontal Sobel filter output, applied to luminance values of the currently selected zone
0x06c	stats_reg_27_hsobel_hi	R		
0x070	stats_reg_28_vsobel_lo	R	0	Higher and lower 32 bits of the sum of absolute values of the Vertical Sobel filter output, applied to luminance values of the currently selected zone
0x074	stats_reg_29_vsobel_hi	R	0	
0x078	stats_reg_30_lsobel_lo	R	0	Higher and lower 32 bits of the sum of absolute values of the Diagonal Sobel filter output, applied to luminance values of the currently selected zone
0x007c	stats_reg_31_lsobel_hi	R	0	
0x0080	stats_reg_32_rsobel_lo	R	0	Higher and lower 32 bits of the sum of absolute values of the anti-diagonal Sobel filter output, applied to luminance values of the currently selected zone
0x084	stats_reg_33_rsobel_hi	R	0	
0x088	stats_reg_34_hifreq_lo	R	0	Higher and lower 32 bits of the sum of absolute values of the High Frequency filter output, applied to luminance values of the currently selected zone
0x08c	stats_reg_35_hifreq_hi	R	0	
0x090	stats_reg_36_lofreq_lo	R	0	Higher and lower 32 bits of the sum of absolute values of the Low Frequency filter output, applied to luminance values of the currently selected zone
0x094	stats_reg_37_lofreq_hi	R	0	
0x098	stats_reg_38_rhist	R	0	Red histogram values calculated over the zones selected by rgb_hist_zone_en
0x09c	stats_reg_39_ghist	R	0	Green histogram values calculated over the zones selected by rgb_hist_zone_en
0x0a0	stats_reg_40_bhist	R	0	Blue histogram values calculated over the zones selected by rgb_hist_zone_en

Table 2-4: **EDK pCore Interface Register Description *(Cont'd)***

Address Offset BASEADDR+	Register Name	Access Type	Default Value	Description
0x0a4	stats_reg_41_yhist	R	0	Luminance histogram values calculated over the zones selected by ycc_hist_zone_en
0x0a8	stats_reg_42_cchist	R	0	Two-dimensional Cr-Cb chrominance histogram values calculated over the zones selected by ycc_hist_zone_en

The core can be effectively reset in-system by asserting `stats_reg01_reset` (bit 0), which returns all register values to their default values. Core outputs are forced to 0 instantaneously until the software reset bit is deasserted. However, block RAMs internal to the core are not initialized until `stats_reg01_sw_reset` is deasserted, and the core becomes ready for the next data-acquisition cycle. For more information on initialization, see Processing States in Chapter 4.

Additional information about programming user registers is provided in the API documentation available in XPS and located in the generated pCore directory under `doc/html/api/index.html` in the EDK pCore Interface, page 14.

Control Register (stats_reg00_control)

The Software Enable bit of register `stats_reg00_control` allows the core to be dynamically enabled or disabled. Disabling the core reduces power consumption when statistical data collection is not needed. The default value of Software Enable is 1 (enabled). See Table 2-5.

Table 2-5: **Control Register**

Position	Name of Flag	Corresponding Event
Bit 0	SW_ENABLE	0: indicates the Image Statistics core is disabled 1: indicates the Image Statistics core is enabled
Bit 1	REG_UPDATE	Semaphore for register update. 0: indicates the Host processor is updating registers 1: indicates the Host processor is done updating registers See the Synchronization, page 33 section for further information.
Bit 2	READOUT	0: directs the Image Statistics core to bypass readout mode. When in readout mode, writing 0 to this flag directs the Image Statistics core to exit readout mode. 1: directs the Image Statistics core to enter readout mode. See the Synchronization, page 33 section for further information.
Bit 3	CLR_STATUS	Resets values of the status register (stats_reg_02_status) to 0, thereby clearing any interrupt requests (irq pin) as well

Bits 1 (REG_UPDATE) and 2 (READOUT) of `stats_reg00_control` provide a frame synchronization mechanism between the EDK processor and the Image Statistics core. For more information on the use of this register, see Synchronization, page 33 Bit 3 (CLR_STATUS) of `stats_reg00_control` provides a mechanism to clear the `Status` register (`stats_reg_02_status`).

Status Register (stats_reg_02_status)

The `status` register contains information about events, such as past timing errors, that the host processor must clear out to be able to detect new or recurring events. See Table 2-6.

Table 2-6: **Status Register**

Position	Name of Flag	Corresponding Event
Bit 0	VSYNC	Falling edge on vblank_in detected
Bit 1	DONE	Frame Data acquisition complete
Bit 2	VBLANK_ERROR	Measured number of total rows per frame is larger than MAX_ROWS parameter
Bit 3	HBLANK_ERROR	Measured number of total columns per frame is larger than MAX_COLS parameter
Bit 4	INIT_DONE	Timing parameters stabilized (goes high after the second frame is completed)
Bits 23-16	VERSION	Core Version number in 5 + 3 bits format. Default value 10h corresponding to version 3.0
Bits 31-29	HISTOGRAM_CONF	These 3 bits indicate whether the core was instantiated with RGB, CC, and Y histograms, respectively, enabled in CORE Generator.

Contents of the `status` register clears with `SCLR` or by asserting `CLR_STATUS` (bit 2 of the `stats_reg00_control` register).

IRQ Control Register (stats_reg02_irq_control)

Once the user application/interrupt handler routine is done servicing the Image Statistics core, the flag that triggered the interrupt should be cleared from software using the `CLR_STATUS` bit of `stats_reg_02_status`, which in turn deasserts the `irq` output pin. See Table 2-7.

Table 2-7: **Interrupt Control Register**

Position	Name of Flag	Description
Bit 0	VSYNC_IRQ_EN	Falling edge on vblank_in (VSYNC) event interrupt enable
Bit 1	DONE_IRQ_EN	Frame Data acquisition complete (DONE) event interrupt enable
Bit 2	VBLANK_IRQ_EN	VBLANK_ERROR event interrupt enable
Bit 3	HBLANK_IRQ_EN	HBLANK_ERROR event interrupt enable
Bit 8	IRQ_EN	General Interrupt Enable

General Purpose Processor Interface

The General Purpose Processor Interface exposes statistical data outputs and all control registers as ports. This option can be used in a system with a user-defined bus interface (decoding logic and register banks) to an arbitrary processor.

The Core Symbol for the General Purpose Processor Interface is shown in Figure 2-2. The Xilinx Streaming Video Interface is described in Table 2-1, and additional ports are described in Table 2-8.

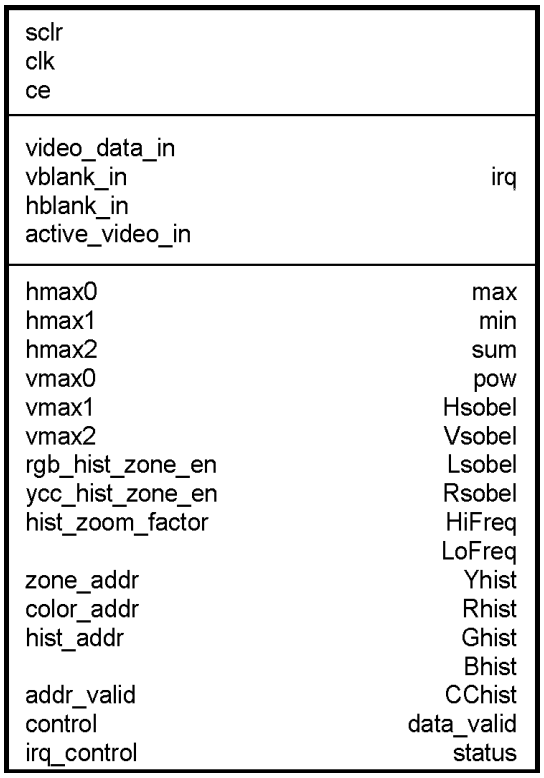


Figure 2-2: **Core I/O Diagram – General Purpose Processor Interface**

To specify the widths of statistical output ports, the following constants are defined:

COLS_WIDTH= floor (log2 (MAX_COLS -1)) +1,

ROWS_WIDTH = floor (log2 (MAX_ROWS-1)) +1,

ROWS_WIDTH = floor (log_2 *(MAX_ROWS))* +1,

SUM_WIDTH= DATA_WIDTH+COLS_WIDTH+ROWS_WIDTH,

SQR_WIDTH = 2DATA_WIDTH+COLS_WIDTH+ROWS_WIDTH,

HIST_WIDTH = COLS_WIDTH+ROWS_WIDTH,

which are at the definitions of input port widths.

Table 2-8: **Ports for the General Purpose Processor Interface**

Signal	Width	Direction	Description
hmax0	COLS_WIDTH	IN	Horizontal coordinate of the first zone delineator
hmax1	COLS_WIDTH	IN	Horizontal coordinate of the second zone delineator
hmax2	COLS_WIDTH	IN	Horizontal coordinate of the third zone delineator
vmax0	ROWS_WIDTH	IN	Vertical coordinate of the first zone delineator
vmax1	ROWS_WIDTH	IN	Vertical coordinate of the second zone delineator
vmax2	ROWS_WIDTH	IN	Vertical coordinate of the third zone delineator
zone_addr	4	IN	During Readout, selects the zone for which max, min, sum, pow, Hsobel and Vsobel values are presented at corresponding outputs
color_addr	2	IN	Selects the color channel (0: Green, 1: Red, 2: Blue) for which max, min, sum, pow, Hsobel and Vsobel values are presented at corresponding outputs
hist_addr	DATA_WIDTH	IN	Address port for reading out histogram values through the Rhist, Ghist, Bhist, Yhist, and CChist outputs
rgb_hist_zone_en	16	IN	Bits 0..15 corresponding to the respective zones control whether the zone is included in RGB histograms
ycc_hist_zone_en	16	IN	Bits 0..15 corresponding to the respective zones control whether the zone is included in Y and 2D Cr-Cb histograms
hist_zoom_factor	2	IN	Values 0,1,2,3 refer to Two-dimensional YCC histogram zooming around the gray point by factors of 1,2,4,8.(1)
addr_valid	1	IN	Logic 1 indicates valid addresses on zone_addr, color_addr and hist_addr
control	4	IN	Bit 0: SW_ENABLE Bit 1: REG_UPDATE Bit 2: READOUT Bit 3: CLR_STATUS
irq_control	9	IN	Bit 0: Falling edge on vblank_in detected (VSYNC) interrupt enable Bit 1: Frame Acquisition done (DONE) interrupt enable Bit 2: Horizontal Framing Error Detected Bit 3: Vertical Framing Error Detected Bit 8: General Interrupt Enabled
max	*DATA_WIDTH*	OUT	Maximum value measured for the zone and color channel selected by zone_addr and color_addr
min	*DATA_WIDTH*	OUT	Minimum value measured for the zone and color channel selected by zone_addr and color_addr
sum	*SUM_WIDTH*	OUT	Sum of values measured for the zone and color channel selected by zone_addr and color_addr
pow	*POW_WIDTH*	OUT	Sum of squares of values measured for the zone and color channel selected by zone_addr and color_addr

Table 2-8: **Ports for the General Purpose Processor Interface *(Cont'd)***

Signal	Width	Direction	Description
hiFreq	*POW_WIDTH*	OUT	Sum of absolute values of the High Frequency filter output, applied to luminance values of the currently selected zone
loFreq	*POW_WIDTH*	OUT	Sum of absolute values of the Low Frequency filter output, applied to luminance values of the currently selected zone.
hsobel	*SUM_WIDTH*	OUT	Sum of luminosity values corresponding to the currently selected zone filtered by a horizontal Sobel Filter
vsobel	*SUM_WIDTH*	OUT	Sum of luminosity values corresponding to the currently selected zone filtered by a vertical Sobel Filter
lsobel	*SUM_WIDTH*	OUT	Sum of luminosity values corresponding to the currently selected zone filtered by a diagonal Sobel Filter
rsobel	*SUM_WIDTH*	OUT	Sum of luminosity values corresponding to the currently selected zone filtered by an anti-diagonal Sobel Filter
rhist	*HIST_WIDTH*	OUT	Red Histogram measurement result corresponding to the current hist_addr address value.
ghist	*HIST_WIDTH*	OUT	Green Histogram measurement result corresponding to the current hist_addr address value
bhist	*HIST_WIDTH*	OUT	Blue Histogram measurement result corresponding to the current hist_addr address value
yhist	*HIST_WIDTH*	OUT	Y (Luminance) Histogram measurement result corresponding to the current hist_addr address value.
cchist	*HIST_WIDTH*	OUT	Two-dimensional CrCb (Chrominance) Histogram measurement result corresponding to the current hist_addr address value
data_valid	1	OUT	Logic 1 indicates valid output on measurement output pins
status	5	OUT	Bit 0: VSYNC falling edge detected Bit 1: DONE: Frame Acquisition Completed Bit 2: VBLANK_error (total rows measured > MAX_ROWS) Bit 3: HBLANK_error (total columns measured > MAX_COLS) Bit 4: INIT_DONE: Timing parameter measurements stabilized
irq	1	OUT	Interrupt request pin
clk	1	input	Rising-edge clock
ce	1	input	Clock enable (active high)
sclr	1	input	Synchronous clear – reset (active high)

1. See Setting Up Histogram Calculations, page 35 for more information.

- **clk - clock:** Master clock in the design, synchronous with, or identical to, the video clock.
- **ce - clock enable:** Pulling `CE` low suspends all operations within the core. Outputs are held, and no input signals are sampled, except for reset (`SCLR` takes precedence over `CE`).
- **sclr - synchronous clear:** Pulling `SCLR` high results in resetting all output pins to zero or their default values. Internal registers within the XtremeDSP™ slice and D-flip-flops are cleared.

Input Registers

Each of the following listed registers are double buffered to prevent the user from inadvertently changing register values while a frame of data is being processed, which could lead to inconsistent measurement results.

- `stats_reg_04_hmax0`
- `stats_reg_05_hmax1`
- `stats_reg_05_hmax2`
- `stats_reg_07_vmax0`
- `stats_reg_08_vmax1`
- `stats_reg_09_vmax2`
- `stats_reg_10_hist_zoom_factor`
- `stats_reg_11_rgb_hist_zone_en`
- `stats_reg_12_ycc_hist_zone_en`

The first set of registers is always available for the host processor to write, while the Image Statistics core is using values from the second set of registers. On frame boundaries (rising edge of `vblank_in`), values from the first set of registers are copied over to the second set of registers if and only if `REG_UPDATE` (bit 1 of the `control` register) is set. This mechanism ensures measurement parameters cannot change while data acquisition is in progress. To avoid using partially updated register values, the host processor should set `REG_UPDATE`=0 before modifying double-buffered input registers, program the registers as needed, then set `REG_UPDATE`=1 to commit changes.

Some controls, such as the `control` register itself, may be modified multiple times mid-frame, so the following registers are not double buffered. Writing into these registers elicits immediate response:

- `stats_reg00_control`
- `stats_reg01_sw_reset`
- `stats_reg03_irq_control`
- `stats_reg13_zone_addr`
- `stats_reg14_color_addr`
- `stats_reg15_hist_addr`
- `stats_reg16_addr_valid`

Chapter 3

Customizing and Generating the Core

This chapter includes information on using Xilinx tools to customize and generate the core.

GUI

The Image Statistics core is easily configured to meet user-specific needs through the CORE Generator graphical user interface (GUI). This section provides a quick reference to the parameters that can be configured at generation time. Figure 3-1 shows the main Image Statistics screen.

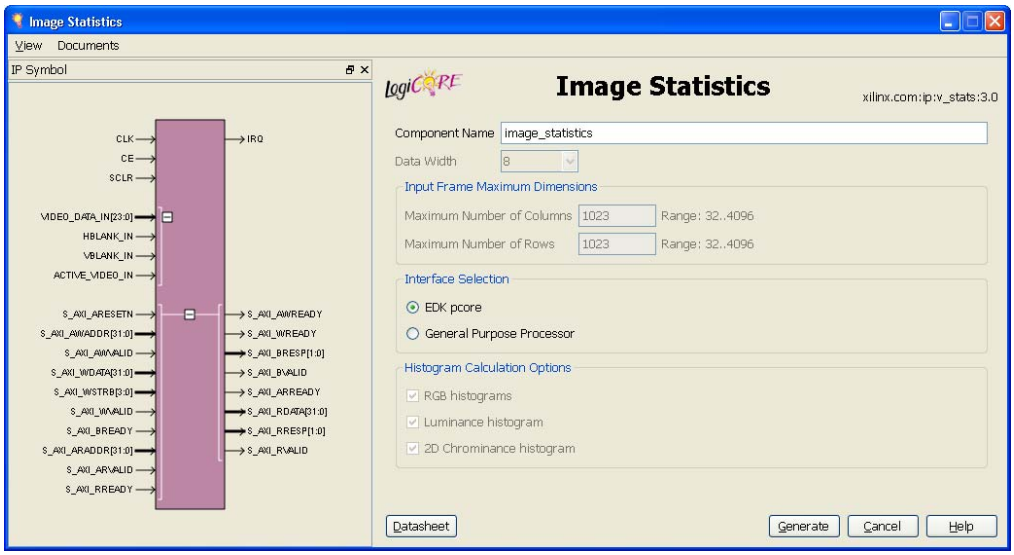


Figure 3-1: **Image Statistics CORE Generator GUI**

The GUI displays a representation of the IP symbol on the left side, and the parameter assignments on the right side, described as follows:

- **Component Name:** The component name is used as the base name of output files generated for the module. Names must begin with a letter and must be composed from characters a to z, 0 to 9 and "_".
- **Data Width** (*DATA_WIDTH*)**:** Specifies the bit width of input data. Permitted values are 8, 10, or 12.
- **Maximum Number of Columns** (*MAX_COLS*)**:** Specifies the number of total columns in a frame defined by the input timing signals. This value is used to determine the depth of line-buffers and the width of certain control paths in the core.

- **Maximum Number of Rows** (*MAX_ROWS*)**:** Specifies the number of total rows in a frame defined by the input timing signals. This value is used to determine the width of certain control paths in the core.
- **Interface Selection:** This option allows for the configuration of two different interfaces for the core.
  - **EDK pCore Interface:** CORE Generator software generates a pCore that can easily be imported into an EDK project as a hardware peripheral. Configuration parameters and statistical data can be accessed via registers. See the EDK pCore Interface in Chapter 2.
  - **General Purpose Processor Interface:** CORE Generator software generates a set of ports to be used to program the core and collect results. See the General Purpose Processor Interface in Chapter 2.
- **Histogram Calculation Options:** Storing and calculating histograms utilize block RAM resources in the FPGA. By specifying which histograms calculations are needed, this option can be used to reduce FPGA resources required for the generated core instance.
  - **RGB Histograms:** The check box enables/disables instantiation of the R,G and B histogram calculating modules for zones pre-selected for RGB histogramming.
  - **Luminance Histogram:** The check box enables/disables instantiation of the luminance histogram calculating module for zones pre-selected for Y and CrCb histogramming.
  - **2D Chrominance Histogram:** The check box enables/disables instantiation of the two-dimensional chrominance (Cr-Cb) histogram calculating module for zones pre-selected for Y and CrCb histogramming

Parameter Values in the XCO File

Table 3-1 shows the possible parameter values and defaults.

Table 3-1: **Parameter Values**

XCO Parameter	Default	Valid Values
chrominance_histogram	true	false, true
component_name	component_name	ASCII text using characters: a..z, 0..9 and "_" starting with a letter. ***Note:*** "v_stats_v3_0" is not allowed.
data_width	8	8, 10, 12
interface_selection	EDK_Pcore	EDK_Pcore, General_Purpose_Processor
luminance_histogram	true	false, true
max_cols	1023	32 - 4096
max_rows	1023	32 - 4096
rgb_histogram	true	false, true

Output Generation

The output files generated from the Xilinx CORE Generator software for the Image Statistics core depend upon whether the interface selection is set to EDK pCore or General Purpose Processor. The output files are placed in CORE Generator's project directory.

EDK pCore Files

When the interface selection is set to EDK pCore, CORE Generator outputs the core as a pCore that can be easily incorporated into an EDK project. The pCore output consists of a hardware pCore and a software driver. The pCore has the following directory structure in the `<project_directory>/<component_name>` folder:

- drivers
  - stats_v3_00_a
    - build
    - data
    - example
    - src
- pcores
  - axi_stats_v3_00_a
    - data
    - hdl
      - vhdl

<project_directory>

This is the top-level directory. It contains xco and other assorted files.

- `<component_name>.xco`: Log file from CORE Generator software describing which options were used to generate the core. An XCO file can also be used as an input to the CORE Generator software.
- `<component_name>_flist.txt`: A text file listing all of the output files produced when the customized core was generated in the CORE Generator software.
- `<component_name>_readme.txt`: A text file listing all of the output files produced and their purposes when the customized core was generated in the CORE Generator software.

<component_name>/pcores/axi_stats_v3_00_a/data

This directory contains files that EDK uses to define the interface to the pCore.

<component_name>/pcores/axi_stats_v3_00_a/hdl/vhdl

This directory contains the Hardware Description Language (HDL) files that implement the pCore.

<component_name>/drivers/stats_v3_00_a/data

This directory contains files that Software Development Kit (SDK) uses to define the operation of the pCore's software driver.

<component_name>/drivers/stats_v3_00_a/src

This directory contains the source code of the pCore's software driver.

- `stats.c`: Provides the Application Program Interface (API) access to all features of the Image Statistics device driver.
- `stats.h`: Provides the API access to all features of the Image Statistics device driver.

General Purpose Processor Files

When the interface selection is set to General Purpose Processor, CORE Generator then outputs the core as a netlist that can be inserted into a processor interface wrapper or instantiated directly in an HDL design. The output is placed in the <project directory>.

The CORE Generator software output consists of some or all the following files:

- `<component_name>_readme.txt`: Readme file for the core.
- `<component_name>.ngc`: The netlist for the core.
- `<component_name>.veo`
- `<component_name>.vho`: The HDL template for instantiating the core.
- `<component_name>.v`
- `<component_name>.vhd`: The structural simulation model for the core. It is used for functionally simulating the core.
- `<component_name>.xco`: Log file from CORE Generator software describing which options were used to generate the core. An XCO file can also be used as an input to the CORE Generator software.
- `<component_name>_flist.txt`: A text file listing all of the output files produced when the customized core was generated in the CORE Generator software.
- `<component_name>.asy`: IP symbol file.
- `<component_name>.gise`
- `<component_name>.xise`: ISE® software subproject files for use when including the core in ISE software designs

Chapter 4

Designing with the Core

This chapter includes guidelines and additional information to make designing with the core easier.

General Design Guidelines

Minimum and Maximum Values

The minimum and maximum values can be useful for histogram stretching, Auto-Gain, Digital-Gain, Auto-Exposure, or simple White-Balance applications. These values are calculated for all zones and all R,G,B color channels simultaneously.

Sum of Color Values

The core provides the sum of color values for all zones (sum). The mean values for color channels can be calculated by dividing the sum value by the size of the zone (*N*):

$$\bar{x} = \frac{1}{N}\sum_{i=0}^{N-1} x_i = \frac{sum}{N}$$

Equation 4-1

Sum of Squares of Color Values

The core provides the power output equal to the sum of squared values for all color channels and zones (*pow*), from which the signal power or the variance can be calculated:

$$\sigma^2 = \frac{1}{N}\left[\sum_{i=0}^{N-1} x_i^2\right] - \bar{x}^2 = \frac{pow}{N} - \bar{x}^2$$

Equation 4-2

Frequency Content

The frequency content for each zone is calculated using the luminance channel. To calculate low-frequency content, luminance values are first low-pass filtered with a 7 tap FIR filter, with fixed coefficients [-1 0 9 16 9 0 -1]/32.

The Low Frequency power output (*LoFreq*) of the core provides the cumulative sum of the squared values of the FIR filter output for each zone:

$$LoFreq = \sum_{i=0}^{N-1} \left[\frac{FIR\left(x_i, \{-1,0,9,16,9,0,-1\}\right)}{32} \right]^2$$

Equation 4-3

In Equation 4-3, square brackets [] represent clipping at $max(x_i)=2^{DATA\_WIDTH-1}$ and clamping values at 0.

The high frequency power output (*HiFreq*) of the core provides the difference between the power of the original luminance values and the power of the low-pass filtered signal within each of these zones:

$$HiFreq = \lfloor pow - LoFreq \rfloor$$

Equation 4-4

In Equation 4-4, square brackets represent clamping values at 0.

Edge Content

The Image Statistics core filters the luminance values calculated for all zones using the Sobel operators:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$$

Figure 4-1: **Horizontal, Vertical and Diagonal (Left and Right) Sobel Operators**

The Sobel operators are implemented without multipliers to reduce size and increase performance. The edge content outputs (*Hsobel*, *Vsobel*, *Lsobel*, *Rsobel*) provide the cumulative sums of absolute values of filtered luminance values:

$$Lsobel = \sum_{i=0}^{N-1} ABS\left(\frac{1}{32} FIR2D\left(x_i, \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \right) \right)$$

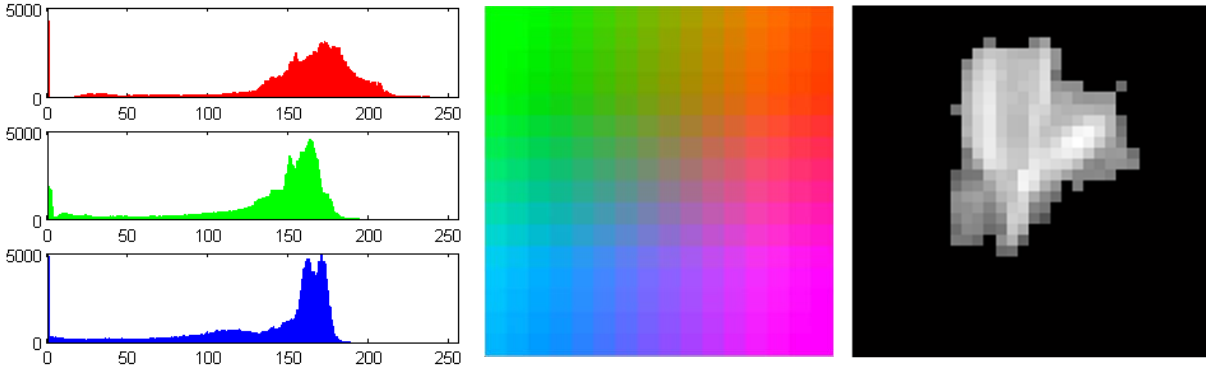
Equation 4-5

Equation 4-5 describes the calculation of the upper-left to lower-right diagonal frequency content, *Lsobel*. Output values for *Vsobel*, *Lsobel*, and *Rsobel* are calculated similarly by using the corresponding coefficient matrixes from Figure 4-1.

Histogram Data

For zones selected by a dynamically programmable register (`rgb_hist_zone_en`), the Image Statistics core bins R,G,B data and creates histograms as shown in Figure 4-2a. Similarly for zones selected by register `ycc_hist_zone_en`, Y and two-dimensional Cr-Cb histograms are calculated as shown in Figure 4-2c.

The two-dimensional Cr-Cb histogram (Figure 4-2c) contains information about the color content of a frame. Different hues have distinct locations in the Cr-Cb color-space (Figure 4-2b). The center location and variance of the color gamut can be derived from its two-dimensional Cr-Cb histogram. The bounding shape of the color gamut, along with the center location and variance of the two-dimensional Cr-Cb histogram, can be used to drive higher level algorithms [Ref 5] for white-balance correction.



a. Example RGB Histograms b. Colors in Cr-Cb Space c. Example Color Gamut in Cr-Cb Space

Figure 4-2: **Histogram Data**

For further details on histogram calculations, refer to Setting Up Histogram Calculations, page 35

The resolution of the R,G,B and Y histograms is the same as the resolution of the input data. The two-dimensional Cr-Cb histogram contains the same number of bins, but due to the two-dimensional configuration, the resolution is $2^{DATA\_WIDTH/2}$ along the Cr-Cb axes.

Initialization

The one- and two-dimensional histograms are stored in block RAMs, which should be cleared before the IP core can start data acquisition. Clearing of block RAMs may take 256, 1024 or 4096 CLK cycles corresponding to 8, 10, or 12-bit wide input data. Block RAM initialization may take several scan-lines, depending on the input resolution.

Once the core is finished with block RAM initialization, it progresses to the “Wait for VBLANK” state where it remains until a falling edge on `vblank_in` is detected, at which time it enters the “Data Acquisition” state.

Data Acquisition

In the “Data Acquisition” state, the core updates all internal measurement values with the pixel data presented on `video_data_in` when data is qualified with `active_video_in` = 1.

The core proceeds to the "Readout" state after the last active scan-line, which may occur several scan-lines before the rising edge on `vblank_in`. The core identifies the last active scan-line based on measurements of the previous frame.

Readout

In the readout state, the core does not collect any more statistical information, and the multiplexing/ addressing mechanism on the output is activated. Once the user provides addresses that are qualified valid by asserting the `addr_valid` pin, the core fetches and displays information on its output ports pertaining to the input addresses. Valid output data is identified by the `data_valid` output pin.

Synchronization

A semaphore-based mechanism is used to synchronize external frame timing with host processor register writes, data readouts and data acquisition.

The semaphores involved in synchronizing host processor activity with the incoming video stream are:

- `DONE` flag (bit 1 of the `status` register)
- `REG_UPDATE` (bit 1 of the `control` register)
- `READOUT` (bit 2 of the `control` register)
- `CLR_STATUS` (bit 3 of the `control` register)

The software flow diagram for normal system operation is shown in Figure 4-3 and is described following the figure.

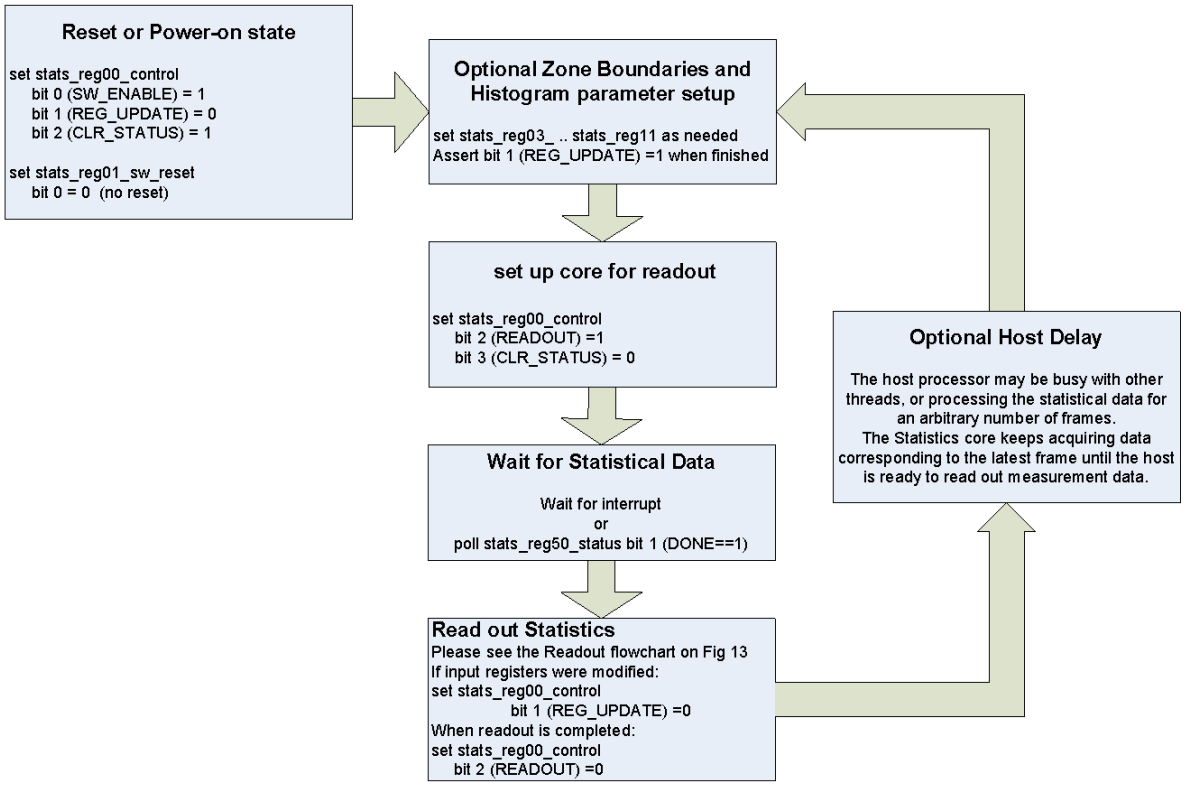


Figure 4-3: **Software Flow Diagram for Normal System Operation**

When data acquisition is finished, the core asserts status flag DONE. If the corresponding Interrupt Enable (DONE_IRQ_EN) and the General Interrupt Enable (IRQ_EN) bits are set to 1, this event also triggers the interrupt request (IRQ) signal.

After the data acquisition state, depending on the state of the READOUT flag, the core either enters the readout state (READOUT = 1), or discards measurement data by re-initializing block RAMs and registers and preparing for acquiring data from the next frame (READOUT = 0).

This mechanism relieves the host processor from having to service the core when statistical data is not needed and allows the core to continuously process frames, so when the host processor is ready to poll statistical information, information from the last frame is available.

If the core enters the readout state, it remains there until the host processor signals being done with reading out measurement data, which may take a few lines, or several frames depending on the speed and the workload of the host processor. Therefore termination of the readout state is decoupled from the input video stream.

Once the host processor deasserts READOUT, the core immediately clears (re-initializes) block RAMs and registers and proceeds to acquire data from the next frame.

After deasserting READOUT, the host processor may assert it again immediately, enabling the core to enter the "Readout" state after acquisition of the current frame is complete.

NOTES:

1. READOUT has to be asserted before the statistics core is done acquiring the next frame, or the core discards the data and self-triggers to acquire the next frame.
2. For the core to process every subsequent frame, the vertical blanking period has to be at least as long as the number of scan-lines it takes to initialize the block RAMs. For example, in the pessimistic case of using SD sensor (720 pixels per line) with 12 bit data (4k deep block RAMs), the minimum vertical blanking period has to be 4096/720 = 5.68, or at least six lines.

Setting Up Zone Boundaries

The zone boundaries for the 16 zones can be set up by programming the positions of three vertical and three horizontal delimiters as shown in Figure 4-4. Complemented by the constraints that the top-left corner of Zone 0 is flush with the left-top corner of the active image, and the bottom-right corner of Zone 15 is flush with the bottom-right corner of the active image, these values uniquely define the corners of all zones.

Data is collected during the active (and non-blank) period of the frame, and all zones traversed by the current scan-line are updated with the input data in parallel. Zone boundaries should be set up before acquiring the first frame of data by programming the hmax and vmax registers.

NOTE: **The minimum horizontal and vertical size of each zone must be at least 2**, along with the following geometric constraints:

- 0 < hmax0 < hmax1 < hmax2 < *MAX_COLS*
- 0 < vmax0 < vmax1 < vmax2 < *MAX_ROWS*

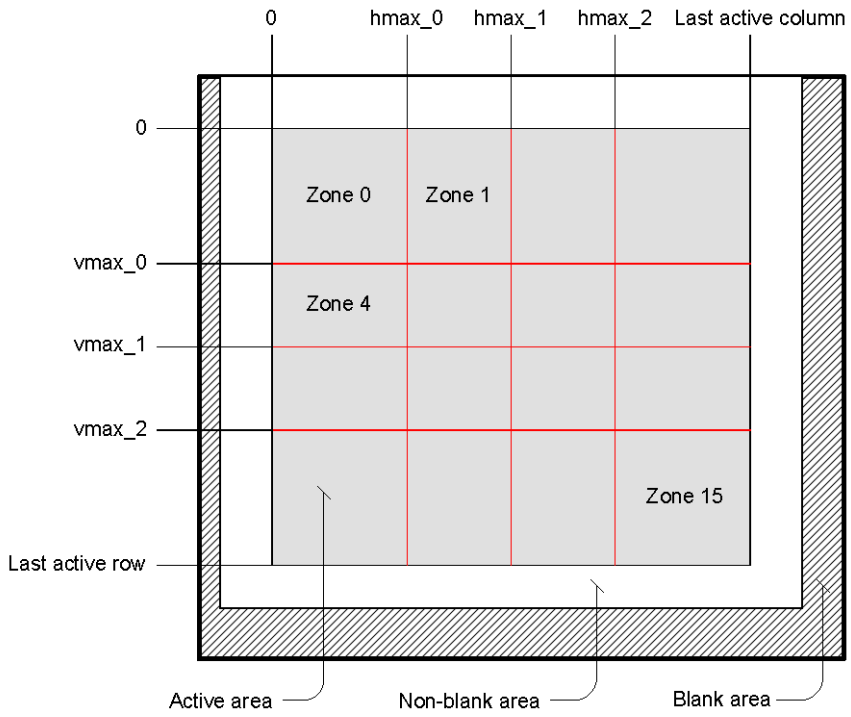


Figure 4-4: **Setting Up Zone Boundaries**

Setting Up Histogram Calculations

Histogram data is calculated and stored in block RAMs; hence calculating RGB and YCrCb histograms for all zones independently significantly increases the amount of FPGA resources required. Employing a mask to select which zones are involved in histogram calculation covers most typical applications:

- Calculating histogram values for one particular zone
- Calculating histogram values over an area of the image (such as the central zones, or zones in the corners)
- Calculating histogram values over the whole image

Figure 4-5 demonstrates how zones for RGB (red squares) and YCrCb (yellow circles) histogram calculations can be selected. For the example shown in Figure 4-5, zones 3, 4, 6, 7, 8 and 14 are selected for the Y and CrCb histograms. Correspondingly, bits 3,4,6,7,8 and 14 are set in `ycc_hist_zone_en`, resulting in a value of 0x000041D8.

Similarly, for the R,G, and B histograms, zones 1, 2, 3, 7, 8, 10, 11 and 14 are selected. Correspondingly bits 1, 2, 3, 7, 8, 10, 11 and 14 are set in `rgb_hist_zone_en`, resulting in a value of 0x00004D8E.

For the two-dimensional Cr-Cb histogram, there is another control, `stats_reg10_hist_zoom_ factor`, that helps tailor the Cr-Cb histogram calculation to the higher-level algorithm that consumes the 2D histogram results.

Consequently, Cr and Cb values have the same dynamic range as the input data. Cr and Cb are represented internally on `DATA_WIDTH` bits. A full precision Cr-Cb histogram would constitute a sparse 4k x 4k table that may be too large to implement within an FPGA. Therefore Cr and Cb are quantized to `DATA_WIDTH/2` bits for histogramming. This quantization process inevitably involves loss of information. The use of the zoom factor

(`stats_reg10_hist_zoom_factor`) enables focusing on certain aspects of the histogram to minimize the effects of the information loss.

Figure 4-5: **Selecting Individual Zones for RGB and YCrCb Histograms**

Some higher level algorithms, such as gamut stretching [Ref 5], are concerned with the overall histogram, while other methods are concerned only with the central section, the area around the neutral point, to identify color casts. To support either type of algorithm, the histogram zoom factor allows the user to trade off resolution with range. The histogram zoom factor controls which bits of Cr and Cb values are selected for histogram binning. By setting the `hist_zoom_factor` to 0, the whole Cr-Cb histogram is represented at the output, as Cr and Cb values are simply quantized to `DATA_WIDTH/2` bits. For example if `DATA_WIDTH` = 8, this quantization results in only the most significant four bits, bits 4, 5, 6 and 7, being used for histogram binning (see Table 4-1).

Table 4-1: **Histogram Zoom Bit Selections for DATA_WIDTH = 8 Bit Input Data**

Histogram Zoom Factor	Bits Used for Binning							
0	7	6	5	4	3	2	1	0
1	7	6	5	4	3	2	1	0
2	7	6	5	4	3	2	1	0
3	7	6	5	4	3	2	1	0

When `hist_zoom_factor` (or `stats_reg10_hist_zoom_factor`) is set to a value other than 0, the resulting two-dimensional histogram represents only the central portion of the Cr-Cb histogram; pixels with extreme Cr-Cb values may fall outside the range represented by `DATA_WIDTH/2` bits.

To enable further reduction of core footprint, RGB, Y, and Cr-Cb histograms can be individually enabled/disabled during generation time via the CORE Generator graphical

user interface. If a particular type of histogram is not needed by the higher level algorithms, the core footprint can be reduced by 1,2, or 4 block RAMs depending on the input data resolution (`DATA_WIDTH`) and the target family.

Processing States

The core distinguishes acquisition and readout periods to avoid modification of single-buffered measurement data while it is being read out. After readout, block RAMs and registers have to be re-initialized before the next acquisition cycle may commence.

After reset or power-up, the core cycles through the "Initialization," "Wait for VBLANK," and "Data Acquisition" states.

Figure 4-6 shows the top-level state diagram of the Image Statistics core.

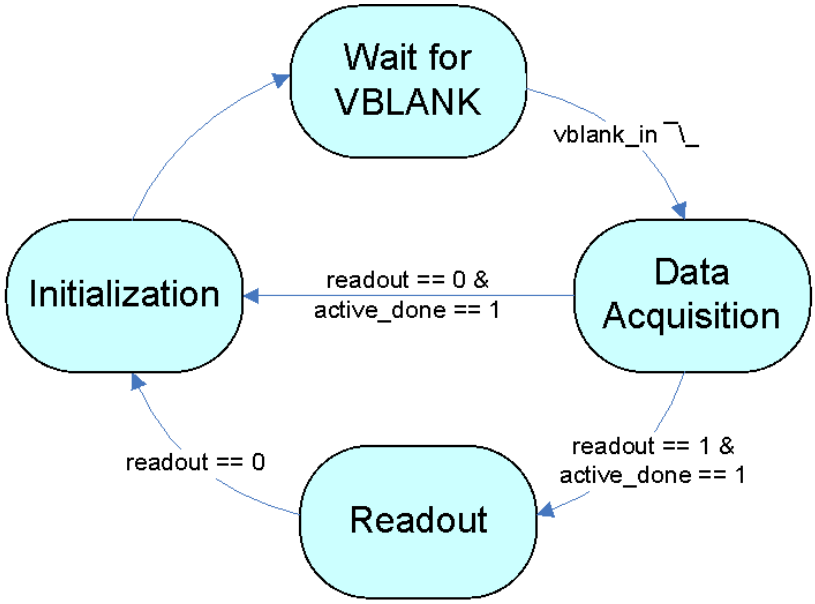


Figure 4-6: **Image Statistics IP Core State Diagram**

Reading Out Statistical Results

Figure 4-7 shows an example of data readout timing and use of the control (specifically bits `REG_UPADTE` and `READOUT`), `irq_control` and `status` (shown in unsigned decimal format) registers. In this example, an empty frame followed by a test frame (`vblank_in`, `hblank_in`, `active_video_in`, `video_data_in`) are processed by the core. The core cycles through the "Clear RAMs" and "Wait on VBLANK" stages, from which it transitions to the "Acquisition" state on the second falling edge of `vblank_in`.

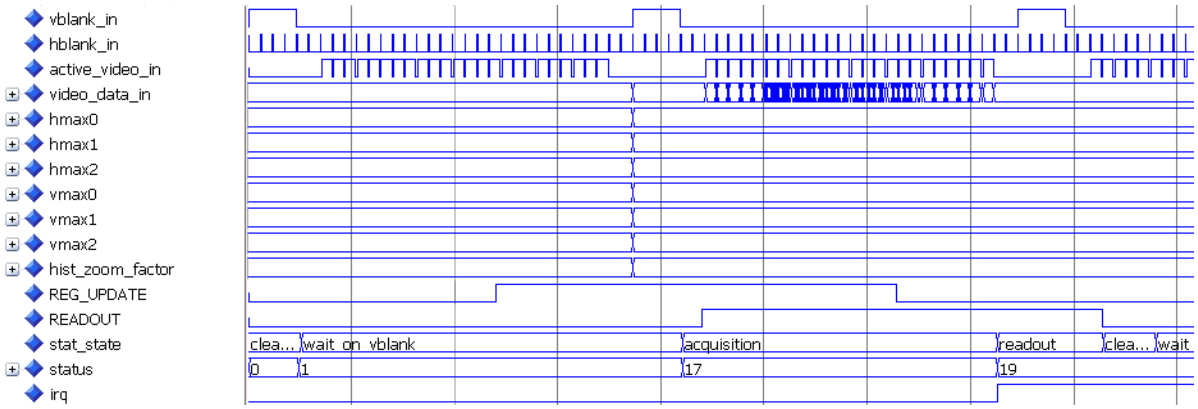


Figure 4-7: **Frame and Readout Timing**

As discussed in the Synchronization, page 33 section, the Image Statistics core signals the end of data acquisition by asserting the DONE flag of the status register, which also triggers an interrupt if the irq_control register is set up to enable interrupts. In this example, bits 8 (IRQ_EN) and 1 (DONE_IRQ_EN) are set to 1, as indicated by the decimal value 258, resulting in the interrupt output (irq) transitioning high (event marked by the red cursor) at the same time the core signals the end of data acquisition (DONE flag of the status register).

During the frame, bit 2 (READOUT) was asserted, which at the end of the active portion of the frame instructs the core to enter the "Readout" state.

Bit READOUT of the control register has to be set before the end of the frame; otherwise the core does not enter the readout mode but clears measurement data and arms itself for capturing the next frame.

After the host processor is finished reading out relevant statistical data, it programs bit 2 (READOUT) of the control register to 0, which instructs the core to re-initialize by entering the "clear-RAMs" state. The example in Figure 4-7 also demonstrates the use of the REG_UPDATE flag. The user at any point could have modified values for input registers, such as hmax0, hmax1, hmax2, vmax0, vmax1, vmax2, rgb_hist_zone_en, ycc_hist_zone_en, or hist_zoom_factor. After setting all input registers to their desired value, REG_UPDATE was asserted, which resulted to all internal registers to latch in the user input at the rising edge of vblank_in. Signals hmax, vmax, rgb_hist_zone_en, ycc_hist_zone_en, and hist_zoom_factor values displayed in Figure 4-7 demonstrate how the internal register values change simultaneously on the rising edge of vblank_in if REG_UPDATE is set to 1.

Addressing

Due to the large number of statistical data collected by the core, presenting all data simultaneously on core outputs is not feasible.

Inputs zone_addr and color_addr for the General Purpose Processor Interface, or registers stats_reg12_zone_addr and stats_reg13_color_addr for the EDK pCore Interface, facilitate reading out the max, min, sum and power result for specific zones and color channels.

The input hist_addr for the General Purpose Processor Interface, or the stats_reg14_hist_addr register for the EDK pCore Interface, facilitate addressing of histogram values.

The Image Statistics core provides a simple handshaking interface for reading out data. After setting the address inputs (registers in case the EDK pCore interface is used) as needed, asserting the `addr_valid` pin signals to the core that valid addresses are present. In turn, the core fetches data corresponding to the addresses and marks valid data on the core outputs by asserting the `data_valid` output/register (Figure 4-8).

All maximum, minimum, sum, sum of squares, Sobel and frequency contents can be read out by accessing zones 0-15 and color channels (coded 0,1,2) sequentially. If the host processor interface and the Image Statistics core are in the same CLK domain, addressing can be simplified, such that multiple addresses are supplied during the active portion of `addr_valid`. When a sequence of valid addresses is presented to the core, the sequence of corresponding valid data becomes available with a latency of five CLK cycles (Figure 4-9).

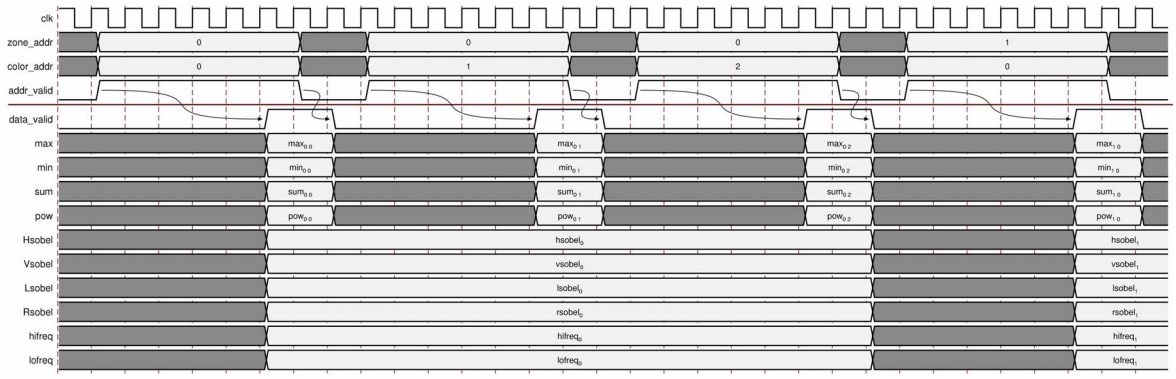


Figure 4-8: **Readout Addressing**

Figure 4-9 illustrates reading out histogram data. To shorten the readout period, histogram data can be read out in parallel with other statistical data.

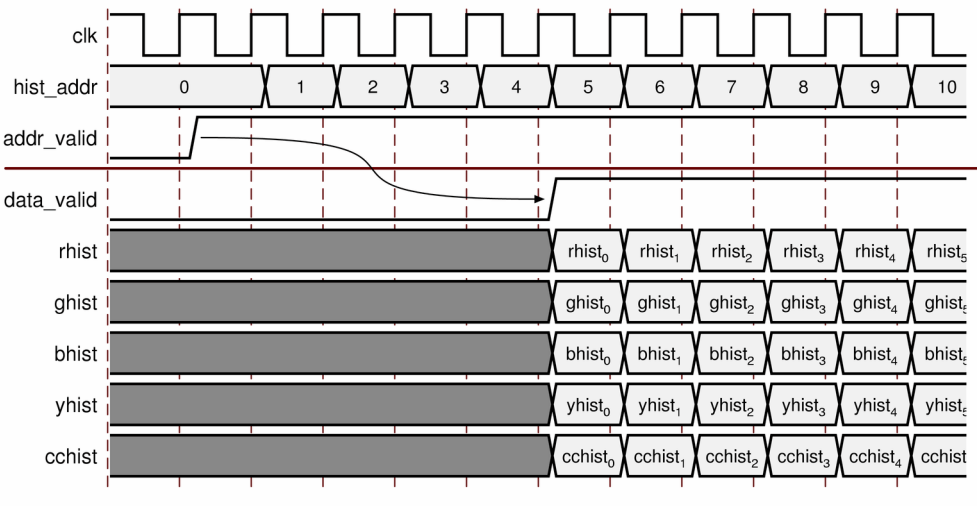


Figure 4-9: **Reading Out Histogram Data**

Clocking

The Image Statistics core has one clock (`clk`) that is used to clock the entire core. This includes the AXI4-Lite interface and the core logic

Resets

The Image Statistics core has one reset (`sclr`) that is used for the entire core. The reset is active High.

Protocol Description

For the pCore version of the Image Statistics core, the register interface is compliant with the AXI4-Lite interface.

Constraining the Core

This chapter contains information on applicable constraints for the Image Statistics core.

Required Constraints

The clk pin should be constrained at the pixel clock rate desired for the video stream.

Device, Package, and Speed Grade Selections

There are no device, package or speed grade requirements for the Image Statistics core. The core has not been characterized for use in low power devices."

Clock Frequencies

The clock for the Image Statistics core should be run at the required pixel clock frequency.

Clock Management

There is only one clock for the Image Statistics core.

Clock Placement

There are no specific clock placement requirements for the Image Statistics core.

Banking

There are no specific banking rules for the Image Statistics core.

Transceiver Placement

There are no transceiver placement requirements for the Image Statistics core.

I/O Standard and Placement

There are no specific I/O standards and placement requirements for the Image Statistics core.

XILINX®

Chapter 6

Detailed Example Design

This chapter describes the example design and associated files.

Directory and File Contents

The directory structure in the top-level folder is as follows:

- Expected: Contains the pre-generated expected/golden data used by the test bench to compare actual output data.
- Stimuli: Contains the pre-generated input data used by the test bench to stimulate the core (including register programming values).
- Results: Actual output data will be written to a file in this folder.
- src: Contains the .vhd & .xco files of the core. The .vhd file is a netlist generated using CORE Generator. A new netlist can be regenerated using the .xco file in CORE Generator.
- tb_src: Contains the top-level test bench design. This directory also contains other packages used by the test bench.
- isim_wave.wcfg: Waveform configuration file for ISim.
- mti_wave.do: Waveform configuration file for ModelSim.
- run_isim.bat: Runscript for ISim in Windows OS.
- run_isim.sh: Runscript for ISim in Linux OS.
- run_mti.bat: Runscript for ModelSim in Windows OS.
- run_mti.sh: Runscript for ModelSim in Linux OS.

Example Design

Demonstration Test Bench

The demonstration test bench is provided as an introductory package that enables core users to observe the core generated by the CORE Generator tool operating in a waveform simulator. The user is encouraged to observe core-specific aspects in the waveform, make simple modifications to the test conditions, and observe the changes in the waveform.

Implementation

Simulation

To start a simulation using ModelSim for Linux, type source **`run_mti.sh`**.

To start a simulation using ModelSim for Windows, double-click on `run_mti.bat`.

To start a simulation using iSim for Linux, double-click on `run_isim.bat`.

Messages and Warnings

Appendix A

Verification, Compliance, and Interoperability

The Image Statistics core has been verified with extensive simulation and hardware verification.

Simulation

A highly parameterizable test bench was used to test the Image Statistics core. Testing included the following:

- Register accesses
- Processing of multiple frames of data
- Testing of various frame sizes
- Varying parameter settings

Hardware Testing

The Image Statistics core has been tested in a variety of hardware platforms at Xilinx to represent a variety of parameterizations, including the following:

- A test design was developed for the core that incorporated a MicroBlaze™ processor, AXI4-Lite interconnect and various other peripherals. The software for the test system included pre-generated input and output data along with live video stream. The MicroBlaze processor was responsible for:
  - Initializing the appropriate input and output buffers
  - Initializing the Image Statistics core.
  - Launching the test.
  - Comparing the output of the core against the expected results.
  - Reporting the pass/fail status of the test and any errors that were found.

XILINX®

Appendix B

Migrating

This appendix describes migrating from older versions of the IP to the current IP release.

Migrating to the EDK pCore AXI4-Lite Interface

The Image Statistics v3.0 changed from the PLB processor interface to the EDK pCore AXI4-Lite interface. As a result, all of the PLB-related connections have been replaced with an AXI4-Lite interface. For more information, see:

http://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf

Parameter Changes in the XCO File

There are no parameter changes in the XCO file.

Port Changes

Other than an AXI4-Lite interface in place of the PLB, there are no port changes.

Functionality Changes

There are no functionality changes to the core.

Special Considerations when Migrating to AXI

The Image Statistics core v3.0 changed from the PLB EDK pCore processor interface to the EDK pCore AXI4-Lite interface. As a result, all of the PLB-related connections have been replaced with an AXI4-Lite interface. This processor interface change does not change the functionality of the core other than an AXI4-Lite interface has to be used in place of the PLB. For more information about AXI4-Lite, see UG761, *Xilinx AXI Reference Guide*.

XILINX®

Appendix C

Debugging

The following questions may help in debugging the core:

- Are the input and output timing signals `active_video`, `vblank`, and `hblank` connected?
- Are the video clock (`clk`) and reset (`sclr`) signals connected?
- Do the control register, status register 1, status register 2 and status register 3 match the default values listed in Table 2-4, page 17?
- Does the software flow look like the software flow diagram listed in Figure 4-3, page 33?
- Did the core timeout? See Full System Hardware Evaluation in Chapter 1 for information on evaluation core timeout.
-

XILINX®

Appendix D

Application Software Development

EDK pCore API Functions

This section describes the functions included in the C driver (`stats.c` and `stats.h`) generated for the EDK pCore API.

The software API is provided to allow easy access to the pCore registers of the Image Statistics IP pCore defined in Table 2-1. To utilize the API functions provided, the following two header files must be included in the user C code:

```
#include “stats.h”
```

```
#include “xparameters.h”
```

The hardware settings of your system, including the base address of your Image Statistics core, are defined in the `xparameters.h` file. The `stats.h` file contains the macro function definitions for controlling the Image Statistics pCore.

The drivers subdirectory of the pCore contains a file, `example.c`, in the `stats_v3_00_a/example` subfolder. This file is a sample C program that demonstrates how to use the Image Statistics pCore API.

Each software register defined in Table 2-4 has a constant defined in `stats.h` that is set as the offset for that register. To write to a register, use the STATS_WriteReg() function using the base address of the Image Statistics pCore instance (from `xparameters.h`), the offset of the desired register, and the data to write.

The definition of this macro is:

STATS_WriteReg(uint32 BaseAddress, uint32 RegOffset, uint32 Data)

This macro writes a given register.

BaseAddress is the Xilinx EDK base address of the Image Statistics core (from `xparameters.h`).

RegOffset is the register offset of the register (defined in Table 2-4).

Data is the 32-bit value to write to the register.

Example:

```
STATS_WriteReg(XPAR_STATS_0_BASEADDR, STATS_REG_00_CONTROL, 1);
```

Similarly, reading a value from a register uses the base address and offset for the register:

STATS_ReadReg(uint32 BaseAddress, uint32 RegOffset)

This macro returns the 32-bit unsigned integer value of the register.

BaseAddress is the Xilinx EDK base address of the Image Statistics core (from xparameters.h).

RegOffset is the register offset of the register (defined in Table 2-1).

Example:

```
Xuint32 value = STATS_ReadReg(XPAR_STATS_0_BASEADDR,
STATS_REG_01_STATUS);
```

Based on the register read and write primitives, the following macros are defined to control the operation of the Image Statistics IP pCore:

STATS_Enable(uint32 BaseAddress)

This macro enables an Image Statistics pCore instance.

BaseAddress is the Xilinx EDK base address of the Image Statistics core (from xparameters.h).

STATS_Disable(uint32 BaseAddress)

This macro disables an Image Statistics pCore instance.

BaseAddress is the Xilinx EDK base address of the Image Statistics core (from `xparameters.h`).

STATS_Reset(uint32 BaseAddress);

This macro resets an Image Statistics instance.

Reset affects the all core measurement outputs immediately, and forcing outputs to 0 until `STATS_ClearReset()` is called.

BaseAddress is the Xilinx EDK base address of the Image Statistics core (from `xparameters.h`).

STATS_ClearReset(uint32 BaseAddress);

This macro clears the reset flag of the core, which allows it to re-sync with the input video stream and return to normal operation.

BaseAddress is the Xilinx EDK base address of the Image Statistics core (from `xparameters.h`).

STATS_RegUpdateEnable(uint32 BaseAddress);

The zone boundary, histogram zone enablement, and histogram zoom factor registers are double-buffered inside the Image Processing core. The first set of registers is always available for the host processor to write, while the Image Statistics core is using values from the second set of registers. On frame boundaries, at the rising edge of `VBlank_in`, values from the first set of registers are copied over to the second set of registers only if `REG_UPDATE` (bit 1 of the control register) is set. This mechanism ensures that measurement parameters cannot change while data acquisition is in progress (for more information see, section Input Registers in Chapter 2).

After updating register values, calling RegUpdateEnable causes the Image Statistics pCore to start using the updated values on the next rising edge of `VBlank_in`. The user must manually disable the register update before register updates begin to make sure all updates will affect the same frame.

This function only works when the Image Statistics core is enabled.

BaseAddress is the Xilinx EDK base address of the Image Statistics core (from `xparameters.h`).

STATS_RegUpdateDisable(uint32 BaseAddress);

The zone boundary, histogram zone enablement, and histogram zoom factor registers are double-buffered inside the Image Processing core. The first set of registers is always available for the host processor to write, while the Image Statistics core is using values from the second set of registers (for more information, see Register Space in Chapter 2).

Disabling the Register Update prevents the Image Statistics pCore to use the freshly updated zone boundary, histogram zone enablement, or histogram zoom factor register values after rising `VBlank_in` edges.

Xilinx recommends that the Register Update be disabled while writing to the zone boundary, histogram zone enablement, and histogram zoom factor registers, until all register write operations are complete.

This function only works when the Image Statistics core is enabled.

BaseAddress is the Xilinx EDK base address of the Image Statistics core (from `xparameters.h`).

XILINX®

Appendix E

C Model Reference

The Image Statistics core has a bit-accurate C model designed for system modeling. The C model features include:

- Bit accurate with Image Statistics core
- Statically linked library (.lib, .o, .obj)
- Available for 32/64-bit Windows, and 32/64-bit Linux platforms
- Supports all features of the Image Statistics core that affect numerical results
- Designed for rapid integration into a larger system model
- Example C code showing how to use the function is provided

Overview

The Xilinx LogiCORE IP Image Statistics v3.0 has a bit accurate C model for 32-bit Windows and 64-bit Linux platforms. The model has an interface consisting of a set of C functions, which reside in a statically link library (shared library). Full details of the interface are given in Image Statistics v3.0 Bit-Accurate C Model, page 52. An example piece of C code showing how to call the model is provided to demonstrate the use of the C model.

The model is bit accurate, as it produces exactly the same output data as the core on a frame-by-frame basis. However the model is not cycle accurate, as it does not model the core's latency or its interface signals.

The latest version of the model is available for download on the Xilinx LogiCORE IP Image Statistics web page.

Software Requirements

The Image Statistics v3.0 C models were compiled and tested with the following software:

Table E-1: **Compilation Tools for the Bit Accurate C models**

Platform	C Compiler
32/64-bit Linux	GCC 4.1.1
32/64-bit Windows	Microsoft Visual Studio 2005

User Instructions

This section contains unpacking and installation instructions for the Image Statistics C model.

Unpacking and Model Contents

Unzip the `v_stats_v3_0_bitacc_model.zip` file, containing the bit accurate models for the Image Statistics IP Core. This produces the directory structure and files shown in Table E-2.

Table E-2: **C Model Directory Structure and Contents**

Name	Description
/lin	Pre-compiled bit accurate ANSI C reference model for simulation on 32-bit Linux Platforms
libIp_v_stats_v3_0_bitacc_cmodel.lib	Image Statistics v3.0 model shared object library (Linux platforms only)
libstlport.so.5.1	STL library, referenced by the Image Statistics and RGB to YCrCb object libraries (Linux platforms only)
run_bitacc_cmodel	Pre-compiled bit accurate executable for simulation on 32-bit Linux Platforms
/lin64	Pre-compiled bit accurate ANSI C reference model for simulation on 64-bit Linux Platforms
libIp_v_stats_v3_0_bitacc_cmodel.lib	Image Statistics v3.0 model shared object library (Linux platforms only)
libstlport.so.5.1	STL library, referenced by the Image Statistics and RGB to YCrCb object libraries (Linux platforms only)
run_bitacc_cmodel	Pre-compiled bit accurate executable for simulation on 32-bit Linux Platforms
/nt	Pre-compiled bit accurate ANSI C reference model for simulation on 32-bit Windows Platforms
libIp_v_stats_v3_0_bitacc_cmodel.lib	Pre-compiled library file for win32 compilation (Windows platforms only)
run_bitacc_cmodel.exe	Pre-compiled bit accurate executable for simulation on 32-bit Windows Platforms
/nt64	Pre-compiled bit accurate ANSI C reference model for simulation on 64-bit Windows Platforms
libIp_v_stats_v3_0_bitacc_cmodel.lib	Pre-compiled library file for win32 compilation (Windows platforms only)
run_bitacc_cmodel.exe	Pre-compiled bit accurate executable for simulation on 64-bit Windows Platforms
README.txt	Release notes
pg008_v_stats.pdf	*LogiCORE IP Image Statistics Product Guide*
v_stats_v3_0_bitacc_cmodel.h	Model header file
run_bittacc_cmodel.c	Example code calling the C model

Table E-2: **C Model Directory Structure and Contents *(Cont'd)***

Name	Description
rgb_utils.h	Header file declaring the RGB image / video container type and support functions
bmp_utils.h	Header file declaring the bitmap (.bmp) image file I/O functions
video_utils.h	Header file declaring the generalized image / video container type, I/O, and support functions
Kodim11.bmp	768x512 sample test image from the True-color Kodak test images

Installation

On Linux, ensure that the directory in which the files libIp_v_stats_v3_0_bitacc_cmodel.so and libstlport.so.5.1 are located is in your $LD_LIBRARY_PATH environment variable.

Image Statistics v3.0 Bit-Accurate C Model

The bit-accurate C model is accessed through a set of functions and data structures, declared in the header file v_stats_v3_0_bitacc_cmodel.h.

Before using the model, the structures holding the inputs, generics and output of the Image Statistics instance have to be defined:

```
struct  xilinx_ip_v_stats_v3_0_generics stats_generics;
struct  xilinx_ip_v_stats_v3_0_inputs   stats_inputs;
struct  xilinx_ip_v_stats_v3_0_outputs  stats_outputs;
```

Declaration of the preceding structs can be found in v_stats_v3_0_bitacc_cmodel.h.

The only generic parameter the Image Statistics v3.0 IP Core bit accurate model takes is the DATA_WIDTH, corresponding to the CORE Generator™ software "Data Width" parameter. Allowed values are 8,10 and 12.

Calling `xilinx_ip_v_stats_v3_0_get_default_generics(&stats_generics)` initializes the generics structure with the Image Statistics GUI default DATA_WIDTH value (8).

The structure `stats_inputs` defines the values of run-time parameters and the actual input image. The structure holds the following members:

Table E-3: **Member Variables of the Input Structure**

Type	Name	Function
video_struct	video_in	Holds the input video stream (may contain multiple frames)
int	hmax0	Column index of the first vertical zone delineator (1)
int	hmax1	Column index of the second vertical zone delineator (1)
int	hmax2	Column index of the third vertical zone delineator (1)
int	vmax0	Row index of the first horizontal zone delineator (1)
int	vmax1	Row index of the second horizontal zone delineator (1)
int	vmax2	Row index of the third horizontal zone delineator (1)

Table E-3: **Member Variables of the Input Structure *(Cont'd)***

Type	Name	Function
int	hist_zoom_factor	Controls CrCb histogram zoom around the gray (center point), which can be useful for white-balance algorithms. 0: No zoom, full Cb and Cr range represented (lowest resolution) 1: Zoom by 2 2: Zoom by 4 3: Zoom by 8 (highest resolution around gray point)
int	rgb_hist_zone_en	16 bit value, each bit controlling whether or not the corresponding zone is included in RGB histogram calculation.
int	ycc_hist_zone_en	16 bit value, each bit controlling whether or not the corresponding zone is included in YCC histogram calculation.
int	frame	The input video struct video_in may contain multiple frames. This value identifies which frame in the input video struct will be analyzed.

1. See Figure 4-1 for the definition of zone delineators.

`xilinx_ip_v_stats_v3_0_get_default_inputs(&stats_generics, &stats_inputs)` initializes members of the input structure with the Image Statistics GUI default values.

Note: The `video_in` variable is not initialized, as the initialization depends on the actual test image to be simulated. The next chapter describes the initialization of the `video_in` structure.

Note: Before calling `xilinx_ip_v_stats_v3_0_get_default_inputs()` it is advised to initialize the `video_in` structure by loading an image or set of video frames. The horizontal and vertical delineators are set by default such that the input images are split into zones with identical dimensions.

After the inputs are defined the model can be simulated by calling the function.

```
int xilinx_ip_v_stats_v3_0_bitacc_simulate(
     struct xilinx_ip_v_stats_v3_0_generics* generics,
     struct xilinx_ip_v_stats_v3_0_inputs*   inputs,
     struct xilinx_ip_v_stats_v3_0_outputs*  outputs).
```

Results are provided in the outputs structure, which contains only one member, type `video_struct`.

After the outputs were evaluated and/or saved, dynamically allocated memory for input and output video structures must be released by calling function

```
void xilinx_ip_v_stats_v3_0_destroy(
     struct xilinx_ip_v_stats_v3_0_inputs *input,
     struct xilinx_ip_v_stats_v3_0_outputs *output).
```

Successful execution of all provided functions except for the destroy function return value 0; otherwise a non-zero error code indicates that problems were encountered during function calls.

Image Statistics Input and Output Video Structure

Input images or video streams can be provided to the Image Statistics v3.0 reference model using the video_struct structure, defined in video_utils.h:

```
struct video_struct{
```

```
int       frames, rows, cols, bits_per_component, mode;
uint16*** data[5]; };
```

Table E-4: **Member Variables of the Video Structure**

Member Variable	Designation
Frames	Number of video/image frames in the data structure
Rows	Number of rows per frame Pertaining to the image plane with the most rows and columns, such as the luminance channel for yuv data. Frame dimensions are assumed constant through the all frames of the video stream, however different planes, such as y,u and v may have different dimensions.
Cols	Number of columns per frame Pertaining to the image plane with the most rows and columns, such as the luminance channel for yuv data. Frame dimensions are assumed constant through the all frames of the video stream, however different planes, such as y,u and v may have different dimensions.
bits_per_ component	Number of bits per color channel / component. All image planes are assumed to have the same color/component representation. Maximum number of bits per component is 16.
Mode	Contains information about the designation of data planes. Named constants to be assigned to mode are listed in Table E-5.
data	Set of 5 pointers to 3 dimensional arrays containing data for image planes. data is in 16 bit unsigned integer format accessed as data[plane][frame][row][col]

Table E-5: **Named Constants for Video Modes with Corresponding Planes and Representations**

Mode	Planes	Video Representation
FORMAT_MONO	1	Monochrome – Luminance only.
FORMAT_RGB(1)	3	RGB image / video data
FORMAT_C444	3	444 YUV, or YCrCb image / video data
FORMAT_C422	3	422 format YUV video, (u,v chrominance channels horizontally sub-sampled)
FORMAT_C420	3	420 format YUV video, (u,v sub-sampled both horizontally and vertically)
FORMAT_MONO_M	3	Monochrome (Luminance) video with Motion.
FORMAT_RGBA	4	RGB image / video data with alpha (transparency) channel
FORMAT_C420_M	5	420 YUV video with Motion
FORMAT_C422_M	5	422 YUV video with Motion
FORMAT_C444_M	5	444 YUV video with Motion

Table E-5: **Named Constants for Video Modes with Corresponding Planes and Representations *(Cont'd)***

Mode	Planes	Video Representation
FORMAT_RGBM	5	RGB video with Motion

1. The Image Statistics core supports the FORMAT_RGB mode

Initializing the Image Statistics Input Video Structure

The easiest way to assign stimuli values to the input video structure is to initialize it with an image or video stream. The bmp_util.h and video_util.h header files packaged with the bit accurate C models contain functions to facilitate file I/O.

Bitmap Image Files

The header bmp_utils.h declares functions which help access files in Windows Bitmap format (http://en.wikipedia.org/wiki/BMP_file_format). However, this format limits color depth to a maximum of 8 bits per pixel, and operates on images with 3 planes (R,G,B). Therefore, functions

```
int write_bmp(FILE *outfile, struct rgb8_video_struct *rgb8_video);
int read_bmp(FILE *infile, struct rgb8_video_struct *rgb8_video);
```

operate on arguments type `rgb8_video_struct`, which is defined in rgb_utils.h. Also, both functions support only true-color, non-indexed formats with 24 bits per pixel.

Exchanging data between rgb8_video_struct and general video_struct type frames/videos is facilitated by functions:

```
int copy_rgb8_to_video( struct rgb8_video_struct* rgb8_in,
                        struct video_struct* video_out );

int copy_video_to_rgb8( struct video_struct* video_in,
                        struct rgb8_video_struct* rgb8_out );
```

Note: All image / video manipulation utility functions expect both input and output structures initialized, for example, pointing to a structure that has been allocated in memory, either as static or dynamic variables. Moreover, the input structure must have the dynamically allocated container (data[ ] or r[ ],g[ ],b[ ]) structures already allocated and initialized with the input frame(s). If the output container structure is pre-allocated at the time of the function call, the utility functions verify and throw an error if the output container size does not match the size of the expected output. If the output container structure is not pre-allocated, the utility functions will create the appropriate container to hold results.

Binary Image/Video Files

The header video_utils.h declares functions which help load and save generalized video files in raw, uncompressed format. Functions

```
int read_video( FILE* infile,  struct video_struct* in_video);
int write_video(FILE* outfile, struct video_struct* out_video);
```

effectively serialize the `video_struct` structure. The corresponding file contains a small, plain text header defining, "Mode", "Frames", "Rows", "Columns", and "Bits per Pixel". The plain text header is followed by binary data, 16 bits per component in scan line continuous format. Subsequent frames contain as many component planes as defined by the video mode value selected. Also, the size (rows, columns) of component planes may differ within each frame as defined by the actual video mode selected.

Working with video_struct Containers

Header file video_utils.h define functions to simplify access to video data in `video_struct`.

```
int video_planes_per_mode(int mode);
int video_rows_per_plane(struct video_struct* video, int plane);
int video_cols_per_plane(struct video_struct* video, int plane);
```

Function `video_planes_per_mode` returns the number of component planes defined by the mode variable, as described in Table 6. Functions `video_rows_per_plane` and `video_cols_per_plane` return the number of rows and columns in a given plane of the selected video structure. The following example demonstrates using these functions in conjunction to process all pixels within a video stream stored in variable `in_video`, with the following construct:

```
for (int frame = 0; frame < in_video->frames; frame++) {
  for (int plane = 0; plane < video_planes_per_mode(in_video->mode); plane++) {
    for (int row = 0; row < rows_per_plane(in_video,plane); row++) {
      for (int col = 0; col < cols_per_plane(in_video,plane); col++) {
        // User defined pixel operations on
        // in_video->data[plane][frame][row][col]
      }
    }
  }
}
```

Destroy the Video Structure

Finally, the video structure must be destroyed to free up memory used to store the video structure.

C Model Example Code

An example C file, run_bitacc_cmodel.c, is provided. This demonstrates the steps required to run the model. After following the compilation instructions, you will want to run the example executable.

The executable takes the path/name of the input file and the path/name of the output file as parameters. If invoked with insufficient parameters, the following help message is printed:

```
Usage: run_bitacc_cmodel in_file out_file
            in_file     : path/name of the input  BMP file
            out_file    : path/name of the output TXT file
```

During successful execution a text file, containing the statistical information by zones and color channels, is created. These output values are bit-true to the corresponding IP core output values, addressed by zone_addr, and color_addr. Histogram values are also contained in the resulting text file in a tabulated format.

Compiling with the Image Statistics v3.0 C Model

Linux

To compile the example code, first ensure that the directory in which the files libIp_v_stats_v3_0_bitacc_cmodel.so and libstlport.so.5.1 are located is present in your $LD_LIBRARY_PATH environment variable. These shared libraries are referenced during

the compilation and linking process. Then cd into the directory where the header files, the library files and run_bitacc_cmodel.c were unpacked. The libraries and header files are referenced during the compilation and linking process.

Place the header file and C source file in a single directory. Then in that directory, compile using the GNU C Compiler:

```
gcc -m32 -x c++ ../run_bitacc_cmodel.c ../parsers.c -o
run_bitacc_cmodel -L. -lIp_v_stats_v3_0_bitacc_cmodel -Wl,-rpath,.

gcc –m64 -x c++ ../run_bitacc_cmodel.c ../parsers.c -o
run_bitacc_cmodel -L. -lIp_v_stats_v3_0_bitacc_cmodel -Wl,-rpath,.
```

Windows

Precompiled library v_stats_v3_0_bitacc_cmodel.lib, and top level demonstration code run_bitacc_cmodel.c should be compiled with an ANSI C compliant compiler under Windows. Here an example is presented using Microsoft Visual Studio.

In Visual Studio create a new, empty Win32 Console Application project. As existing items, add:

- `libIpv_stats_v3_0_bitacc_cmodel.lib` to the “Resource Files” folder of the project
- `run_bitacc_cmodel.c` to the “Source Files” folder of the project
- `v_stats_v3_0_bitacc_cmodel.h` header files to “Header Files” folder of the project (optional)

After the project has been created and populated, it needs to be compiled and linked (built) in order to create a win32 executable. To perform the build step, choose “Build Solution” from the Build menu. An executable matching the project name has been created either in the Debug or Release subdirectories under the project location based on whether “Debug” or “Release” has been selected in the “Configuration Manager” under the Build menu.

XILINX®

Appendix F

Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

http://www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:

http://www.xilinx.com/support/documentation/sw_manuals/glossary.pdf.

References

1. AMBA AXI Protocol Version: 2.0 Specification
2. UG761, *Xilinx AXI Design Reference Guide*
3. Vicent Caselles, Jose-Luis Lisani, Jean-Michel Morel, Guillermo Sapiro: *Shape Preserving Local Histogram Modification*
4. Joung-Youn Kim, Lee-Sup Kim, and Seung-Ho Hwang: *An Advanced Contrast Enhancement Using Partially Overlapped Sub-Block Histogram Equalization*
5. Simone Bianco, Francesca Gasparini and Raimondo Schettini: *Combining Strategies for White Balance*
6. G. Finlayson, M. Drew, and B. Funt, "Diagonal Transform Suffice for Color Constancy" in *Proc. IEEE International Conference on Computer Vision*, Berlin, pp. 164-171, 1993
7. Keith Jack: Video Demystified, 4th Edition, *ISBN 0-7506-7822-4*, pp 15-19

Technical Support

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled DO NOT MODIFY.

See the IP Release Notes Guide (XTP025) for more information on this core. For each core, there is a master Answer Record that contains the Release Notes and Known Issues list for the core being used. The following information is listed for each version of the core:

- New Features
- Resolved Issues
- Known Issues

Ordering Information

The Image Statistics core is provided under the Xilinx Core License Agreement and can be generated using the Xilinx® CORE Generator™ system. The CORE Generator system is shipped with Xilinx ISE® Design Suite software.

A simulation evaluation license for the core is shipped with the CORE Generator system. To access the full functionality of the core, including FPGA bitstream generation, a full license must be obtained from Xilinx. For more information, visit the Image Statistics product page.

Contact your local Xilinx sales representative for pricing and availability of additional Xilinx LogiCORE IP modules and software. Information about additional Xilinx LogiCORE IP modules is available on the Xilinx IP Center.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/19/11	1.0	Initial Xilinx release.

Notice of Disclaimer